FINAL EVALUATION REPORT
Wang Government Services, Incorporated
XTS-300 STOP 5.2.E

NATIONAL

COMPUTER SECURITY CENTER

9800 Savage Road
Fort George G. Meade
Maryland 20755-6000

3 August 2000

Report No. CSC-EPL-92/003.E

This page intentionally left blank

FOREWORD

This publication, the **Final Evaluation Report Wang XTS-300** is being issued by the National Computer Security Center under the authority of and in accordance with DoD Directive 5215.1, "Computer Security Evaluation Center." The purpose of this report is to document the results of the formal evaluation of Wang XTS-300 operating system. The requirements stated in this report are taken from **Department of Defense Trusted Computer System Evaluation Criteria** , dated December 1985.

Approved:

---

3 August 2000

Director,
National Computer Security Center

ACKNOWLEDGMENTS

Team members included the following individuals, who were provided by the indicated organizations:

Jerome F. Myers
The Aerospace Corporation
Columbia, Maryland

Scott Barman
Louise Huang
Richard H. Murphy
J. David Thompson
Mitretek Systems
McLean, Virginia

Steve Monaco
National Security Agency
Fort Meade, Maryland

John G. Ata
James E. Knoke
Wang Government Services, Incorporated
Herndon, Virginia

Technical support for the Covert Channel Analysis was provided by:

Dr. Jonathan K. Millen
The MITRE Corporation
Bedford, Massachusetts

Jerome F. Myers
The Aerospace Corporation
Columbia, Maryland

W. Olin Sibert
Oxford Systems, Inc.
Lexington, Massachusetts

The evaluation team acknowledges the great deal of effort put forth by previous evaluation team members, including the production of much of this report:

Daniel P. Faigin
Chao-Hsing Pian
The Aerospace Corporation
Los Angeles, California

James Donndelinger
The Aerospace Corporation
Columbia, Maryland

Dr. Santosh Chokhani
Barbara A. Maguschak
The MITRE Corporation
McLean, Virginia

Frank Belvin
Brett C. Borgeson
Jean-Paul Otin
Shaan Razvi
Harold J. Wolfe
The MITRE Corporation
Bedford, Massachusetts

Heidi K. Henson
John A. Lawrence
National Security Agency
Fort Meade, Maryland

TABLE OF CONTENTS

This page intentionally left blank

FIGURES

This page intentionally left blank

TABLES

This page intentionally left blank

EXECUTIVE SUMMARY

The security protection provided by the Wang Government Services, Incorporated *XTS-300* system, when configured in a secure manner as described in the **XTS-300 Trusted Facility Manual** [33], was evaluated by the National Security Agency (NSA). The security features of XTS-300 were examined against the requirements specified by the **Department of Defense Trusted Computer System Evaluation Criteria** [11] dated December 1985 (TCSEC) to establish a candidate rating. The NSA evaluation team determined that the highest class at which XTS-300 satisfies all the specified requirements of the TCSEC is B3. Therefore, XTS-300, when configured as described in the Trusted Facility Manual, was assigned a Class B3 rating.

A system that is rated as a B3 class system provides a Trusted Computing Base (TCB) that enforces a mandatory and discretionary access control policy. In addition, the TCB provides a trusted path to ensure a reliable TCB-to-user communication connection, and an alarm mechanism to detect the accumulation of events that indicate an imminent violation of the security policy. Separate administrator and operator roles are defined. The least privilege principle was applied in the design of the TCB such that only those functions requiring privileges have them. The TCB was analyzed and found to meet the minimization requirement which reduces complexity, and to meet the layering, abstraction, and data-hiding requirement. The TCB has been tested thoroughly and found to be resistant to penetration. The system developer also provided a model and a descriptive top-level specification on which the design of the TCB is based.

The XTS-300 is hosted on Intel Pentium II/III based server class systems, available in tower and rack-mount form factors. The XTS-300 uses specific Intel-brand motherboards and industry standard ISA and PCI peripheral cards or chips built into the motherboard. Symetric multi-processor (SMP) configurations are supported. Support for SCSI peripherals such as hard disks, CDs, PCMCIA/PC-Card readers and tape drives is provided in the STOP 5.2.E operating system. Support for video including UNIX's X-Windows is provided. Network support for TCP/IP based 10 or 100BaseT networks is provided. Wang has developed STOP 5.2.E, which is a multilevel secure operating system that runs on the XTS-300 hardware. The XTS-300 provides a process virtual memory of up to four gigabytes, and uses the hardware protection level (ring) mechanism in conjunction with software mechanisms for protection.

STOP 5.2.E is a multiprogramming system that can support up to 19 connections and 200 processes. STOP 5.2.E consists of four components: the Security Kernel, which operates in the most privileged ring and provides all mandatory and a portion of the discretionary access control; the TCB System Services (TSS), which operates in the next-most-privileged ring, and implements a hierarchical file system, supports user I/O, and implements the remaining discretionary access control; Trusted Software, which provide the remaining security services and user commands; and the Commodity Application System Services (CASS), which operates in a less privileged ring and provides the UNIX-like interface. CASS is not in the TCB and hence was not thoroughly examined by the evaluation team.

XTS-300 is designed to provide a high level of security for many environments including specialized applications such as Email, FTP, and socket-based database synchronization guards which filter the information according to rules based on the security policy needed by a location. The system provides mandatory and discretionary access control which allows for both a secrecy and integrity policy. The system provides for user identification and authentication used for policy enforcement through user identifiers and passwords, and individual accountability through its auditing capability. Data scavenging is prevented through the control of object reuse. The trusted path mechanism is provided by the implementation of a Secure Attention Key (SAK). The separation of administrator and operator roles is enforced through integrity protected operations.

XTS-300 is marketed and supported by Wang. The evaluated version of the operating system is STOP 5.2.E, which was released in April 2000.

Chapter 1

Introduction

In July 1987, the National Security Agency (NSA) began a product evaluation of XTS-200, a product of HFS Incorporated, now Wang Government Services, Incorporated. The XTS-200, including STOP 3.1.E, was formally evaluated and placed in the Evaluated Products List (EPL) in May of 1992.

In May 1993, a Future Change Review Board (FCRB) meeting was held to discuss the changes planned to the XTS-200 and those planned for XTS-300. A security analysis team was then assigned to evaluate the updated system (XTS-200). The XTS-200, including STOP 3.2.E, was formally evaluated and placed on the EPL in January of 1994.

In September 1994, a security analysis team was assigned to evaluate the changes made for XTS-300. The XTS-300 including STOP 4.1 was then placed on the EPL in July 1995. In October of 1995, STOP 4.1.a, a minor revision, was also placed on the EPL.

In November 1996, a security analysis team was assigned to evaluate the changes made for XTS-300. The XTS-300 including STOP 4.4.2 was then placed on the EPL in March 1998.

In January 1999, a security analysis team was assigned to evaluate the changes made for STOP 5.2.E. The objective of the evaluation was to rate the XTS-300, including STOP 5.2.E, against the **Department of Defense Trusted Computer System Evaluation Criteria** [11] (TCSEC), and to place it on the EPL with a final rating.

This report documents the original results of the evaluation and subsequent analysis, which applies to the system as available from Wang in April 2000.

Material for this report was gathered by the SA team from evaluation evidence for the XTS-300, including STOP 5.2.E, and through documentation, interaction with system developers, and by extensive testing of the system.

1.1 Evaluation Process Overview

The Department of Defense Computer Security Center was established in January 1981 to encourage the widespread availability of trusted computer systems for use by facilities processing classified or other sensitive information. In August 1985 the name of the organization was changed to the National Computer Security Center. In order to assist in assessing the degree of trust one could place in a given computer system, the DoD Trusted Computer System Evaluation Criteria (TCSEC) was written. The TCSEC establishes specific requirements that a computer system must meet in order to achieve a predefined level of trustworthiness. The TCSEC levels are arranged hierarchically into four major divisions of protection, each with certain security-relevant characteristics. These divisions are in turn subdivided into classes. To determine the division and class at which all requirements are met by a system, the system must be evaluated against the TCSEC by an NSA, Trusted Product and Network Security evaluation team.

The NSA supports the creation of secure computer products in varying stages of development from initial design to those that are commercially available. Preliminary to an evaluation, products must go through the Proposal Review Phase. This phase includes an assessment of the vendor's capability to create a secure system and complete the evaluation process. To support this assessment, a Preliminary Technical Review (PTR) of the system is done by the NSA. This consists of a quick review of the current state of the system by a small, but expert, team and the creation of a short report on the state of the system. If a vendor passes the Proposal Review Phase they will enter a support phase preliminary to evaluation. This support phase has two steps, the Vendor Assistance Phase (VAP) and the Design Analysis Phase (DAP). During VAP, the newly assigned team reviews design specifications and answers technical questions that the vendor may have about the ability of the design to meet the requirements. A product will stay in VAP until the vendor's design, design documentation, and other required evidence for the target TCSEC class are complete and the vendor is well into implementation. At that time, the support moves into DAP.

The primary thrust of DAP is an in-depth examination of a manufacturer's design for either a new trusted product or for security enhancements to an existing product. DAP is based on design documentation and information supplied by the industry source, it involves little "hands on" use of the system, but during this phase the vendor should virtually complete implementation of the product. DAP results in the production of an Initial Product Assessment Report (IPAR) by the NSA assessment team. The IPAR documents the team's understanding of the system based on the information presented by the vendor. Because the IPAR contains proprietary information and represents only a preliminary analysis by the NSA, distribution is restricted to the vendor and the NSA.

Products that have completed the support phase with the successful creation of the IPAR, enter formal evaluation. Products entering formal evaluation must be complete security systems. In addition, the release being evaluated must not undergo any additional development. The formal evaluation is an analysis of the hardware and software components of a system, all system documentation, and a mapping of the security features and assurances to the TCSEC. The analysis performed during the formal evaluation requires "hands on" testing (i.e., functional testing and, if applicable, penetration testing). The formal evaluation results in the production of a final report and an Evaluated Products List entry. The final report is a summary of the evaluation and includes the EPL rating which indicates the final class at which the product satisfies all TCSEC requirements in terms of both features and assurances. The final report and EPL entry are made public.

After completion of the Formal evaluation phase, products enter the rating maintenance phase (RAMP). The rating maintenance phase provides a mechanism to extend the previous rating to a new version of an evaluated computer system product. As enhancements are made to the computer product the ratings maintenance phase ensures that the level of trust is not degraded.

Rating Maintenance is accomplished by using qualified vendor personnel to manage the change process of the rated product during the maintenance cycle. These qualified vendor personnel must have strong technical knowledge of computer security and of their computer product. These trained personnel will oversee the vendor's computer product modification process. They will demonstrate to the Trusted Product and Network Security Evaluation Division that any modification or enhancements applied to the product preserve the security mechanisms and maintain the assurances required by the TCSEC for the rating previously awarded to the evaluated product.[1]

[1] The TCSEC evaluation process described here was employed in the initial evaluation of this product and its subsequent RAMPs. The TCSEC process is being replaced by a Common Criteria evaluation process.

1.2 Document Organization

This report consists of nine major sections and four appendices. Section 1 is the introduction. Sections 2 through 7 provide an overview of the system, its hardware and software architecture, and a description of the security support (Trusted Computing Base (TCB) protection mechanisms and assurances). Section 8 provides a mapping between the requirements specified in the TCSEC and the system features that fulfill those requirements. The last section of the main body of the report contains additional comments from the evaluation team about the system. The appendices identify specific hardware and software components to which the evaluation applies, and provide reference information.

Proprietary information has been removed from this version of the Final Evaluation Report, and is replaced in the text by the note: "Proprietary material removed."

This page intentionally left blank

Chapter 2

System Overview

The system described in this report is XTS-300 which includes the STOP 5.2.E operating system and commercially available hardware products. STOP 5.2.E is a product of Wang Government Services, Incorporated. The hardware consists of components manufactured by third-party vendors. The hardware includes the Intel Pentium II/III processor and associated peripherals, including a hard disk, floppy disk drive, SCSI adapter, Ethernet card, streamer tape drive, mouse and keyboard. Other optional peripherals include a parallel printer, the PC Card Reader/Writer, CD-ROM and the Data Transfer Cartridge (DTC) devices. The XTS-300 is assembled, tested, and distributed by Wang.

The system supports both a mandatory sensitivity policy and a mandatory integrity policy. It provides 16 hierarchical sensitivity levels, 64 non-hierarchical sensitivity categories, eight hierarchical integrity levels, and 16 non-hierarchical integrity categories. Some of the hierarchical integrity levels are used by the system to provide role separation, and the others are available to users. The combination of mandatory sensitivity and integrity hierarchical and non-hierarchical levels is called the *Mandatory Access Control (MAC) label*. (In this report, the term *maximum MAC label* denotes a label with the maximum sensitivity and maximum integrity levels, and with all possible non-hierarchical categories.) The system also supports a discretionary access control policy.

XTS-300 consists of two major components: the Trusted Computing Base (TCB), and Commodity Application System Services (CASS). The TCB contains the XTS-300 hardware, with the exception of that connected to the Znyx LAN adapter for purposes of network connections, and the software portion of STOP 5.2.E that is trusted. STOP 5.2.E consists of four components, three of which are in the TCB, the fourth of which is CASS. The TCB provides all basic operating system services and enforces the system security policy, while CASS provides the user with an application programming environment. Although CASS is not part of the TCB, it is discussed in this report because it, or a site-provided alternative, is necessary for the system to support users. The software portion of the TCB consists of a *Security Kernel* that provides basic operating system, I/O functions, and security services, a higher layer called *TCB System Services (TSS)* that provides a file system and other services, and *Trusted Software* that provides functions available to the user, system operator, and system administrator. Wang has chosen to make the user interface of CASS similar to that provided by UNIX, so that it will support applications developed to operate under UNIX.

The network subsystem is host-based and resides in the TCB. It is completely included in the evaluated configuration. Attaching the XTS-300 to a network is permitted in the evaluated configuration. The following summarizes the network security policy that must be in place when a network is attached to an XTS-300:

- Networks are single-level and unlabeled. All data going out a network must be at the same level as the network device and any data coming in from the network will be labeled at that level. The TCB does not attach labels to data going out to a network and does not support lower protocol stack labeling standards such as CIPSO.
- The TCB will generally treat a TCP/IP connection as accessible only to the process which established it. The exceptions to this are that a process may explicitly "register" a socket for sharing with a

single, other process and that a forked child process will have access to sockets that were open in the parent process. The TCB does not maintain DAC attributes for sockets and sockets are not considered "named" objects.

- Remote logins to the TCB, across the network, are not supported and there is no trusted path across the network. No "inbound" high-level protocols (e.g., Telnet and FTP) that involve user authentication are allowed by the TCB.
- No exchange of audit information or information from other trusted databases are allowed across the network.
- Each network device must be configured with a specific owner and the discretionary permissions must be configured to disallow access to any process with a different owner.

For more detailed information on the network security policy, see Section 5 of the **XTS-300 Trusted Facility Manual** [33]

When attaching an XTS-300 to a network, a site should recognize that a network system viewed as a whole is subject to additional security risks that are not present for a stand-alone computer system. Incorporation of an XTS-300 into a network does not eliminate these risks. Some examples of general risks are itemized below.

- An unauthorized person could use a passive wiretap to read data on the network. This data includes Telnet and FTP passwords coming from the XTS-300. If the network is classified, classified data could be disclosed.
- Although the XTS-300 does not allow promiscuous mode to be enabled, or restricts the use of raw IP to privileged users, use of these modes by authorized users on any non-XTS-300 network node effectively bypasses these restrictions and could result in DAC violations or network spoofing.
- Spoofing (i.e., masquerading) a response to the XTS-300 (or any workstation) during an FTP or Telnet session could mislead the user as to the actual data transferred or the actual (remote) operations performed.
- Any network user could disrupt or deny network service or even slow down local workstation operations (even on the XTS-300) by several methods including flooding the network or a particular host with packets.
- Any network user may be able to cause a system on the network to shut down by causing network logs or system audit files to fill.

The XTS-300 TCB software can not itself provide complete countermeasures for the network security concerns mentioned above. Note that the threats are not to XTS-300 security, only to security of the network viewed as a whole. A site will have to rely on physical, personnel, and procedural methods to counter the threats and some such countermeasures are itemized below. Most of these countermeasures are not XTS-300 specific and in these cases no XTS-300 specific procedures for implementing them are given.

- Do not directly connect the XTS, or the internal LANs it connects to, to a Wide Area Network (WAN). Always connect WANs to devices, such as routers, that filter packets. Filter out incoming

ICMP packets, layer 2 packets larger than about 3000 bytes, and incoming packets with a source address that is inside the LAN. This filtering will prevent ICMP redirect attacks and certain kinds of denial-of-service attacks (unless they originate from a system on the LAN) and will make it more difficult for attackers to use IP spoofing.

- Refrain from configuring multiple network interfaces on the XTS at the same MAC level. Configuring network interfaces at different levels may require use of custom, trusted software to allow communications between different networks. However, use of IP forwarding and IP source routing can be nullified, and risk of ICMP echo attacks can be reduced, by separating networks at different MAC levels.
- Network administrators should ensure that each LAN that is hooked up to the XTS-300 operates only at the expected level.
- Establish consistent procedures and policies for all workstations on the network (e.g., disallow use of "raw IP" and "promiscuous mode").
- Enable auditing of ICMP redirects by the XTS.
- Refrain from enabling IP forwarding on the XTS.
- Physically protect the network media and all hardware attached to it.
- A TCP/IP network has privileged ports that are normally restricted from being accessed by user programs. On UNIX systems, this is often accomplished by restricting access to these ports to programs running with the root user ID. On the XTS-300, these privileged ports are restricted by designating a special network user ID. Therefore, the Systems Administrator should ensure that the network user ID is only used for this purpose. It should not be possible to interactively log in as the network user. Finally, the System Administrator should ensure that only appropriate programs are allowed to run with the userid of this user.

For more detailed information on both potential threats and countermeasures, see Section 2 of the **XTS-300 Trusted Facility Manual** [33]

2.1 XTS-300 Background and History

XTS-200, XTS-300, STOP 3.1.E, STOP 3.2.E, STOP 4.1, and STOP 5.2.E are descendants of the Secure Communications Processor (SCOMP) [13] a system also developed by Wang, which was evaluated by the National Computer Security Center (NCSC) in 1984 and received an A1 rating. The hardware base is fundamentally different than the SCOMP's, and the software has also undergone significant further development. Development of STOP 3.1 on the DPS6 PLUS began in 1987 and continued through 1989. During that time, the combination of the software and hardware became known as XTS-300.

In contrast to the hardware on which the SCOMP was based (the DPS6), the DPS6 PLUS and DPS 6000 incorporate virtual memory and ring-protection techniques from the Multics [12] system, so no additional hardware modification was required. The salient differences between the operating system employed by SCOMP and XTS-200 are that STOP 3.1 incorporated complete file system support within the TCB, and that it was considerably restructured to improve its use of layering and other software engineering principles. In addition, XTS-200 supported multiprocessor configurations. Development on STOP 3.2.E began in June 1992 and was completed in June 1993.

The essential differences between XTS-300 and XTS-200 result from the change in hardware base, primarily to employ the functions provided by the Intel processor. Memory management, ring protection logic, and process management have been updated in XTS-300. In addition, I/O functions have been moved from TSS into the kernel. However, the user interface to XTS-300 has changed little from that of XTS-200. Development on STOP 4.1 began in November 1992 and was completed in October 1994. Development continued through STOP 5.2.E which was completed in April 2000.

The overview begins with an examination of the hardware, followed by a description of the software.

Chapter 3

Hardware Overview

3.1 Introduction

The XTS-300 is a 32-bit, demand-paging, time-sharing, single or multi-processor system. Separation of data is accomplished by a combination of hardware and software. The Intel Pentium II and Pentium III, upon which the XTS-300 is based, incorporate their own ring protection mechanism supporting four rings, descriptor privilege levels, gate descriptors, segment attributes (read, write, execute), and call/return instructions. The privilege level (PL) protection mechanism ranges from PL0 (the most privileged) to PL3 (the least privileged). The CPUs support instructions to address bits, bytes, words, and double words (32-bits). They also support up to 256 interrupt vectors, although the XTS-300 does not use all of these. Instruction addressing modes include displacement, base, base plus displacement, scaled index plus displacement, base plus index plus displacement, and base plus scale index plus displacement.

The XTS-300 does not allow loading of firmware by users or processes. The Kernel does however load firmware into the Adaptec AIC7870/96 SCSI controller during system startup. Other hardware components support firmware downloading, but the TCB prevents processes from doing that.[1]

The XTS-300 has a Intel Pentium II/III, PCI bus hardware base. It uses the following Commercial Off-The-Shelf (COTS) peripherals:

- Hard Disk
- Intel Pentium II/III CPU, PCI Bus Motherboard w/on-board: video, SCSI and Parallel/Serial I/O
- up to 512 Mbytes RAM
- 3.5″ 1.44 Mbyte Floppy Disk (capable of reading standard density diskettes)
- SCSI Host Adapter (One to two on-board and optional add-ons)
- 4 mm DAT Tape Drive
- PC Card Reader
- Data Transfer Cartridge Reader
- CD-ROM Drive
- SVGA Monitor
- 101-key, 104-key standard or laptop style Keyboard

[1] The 4mm tape drive has its firmware modified to prevent downloads.

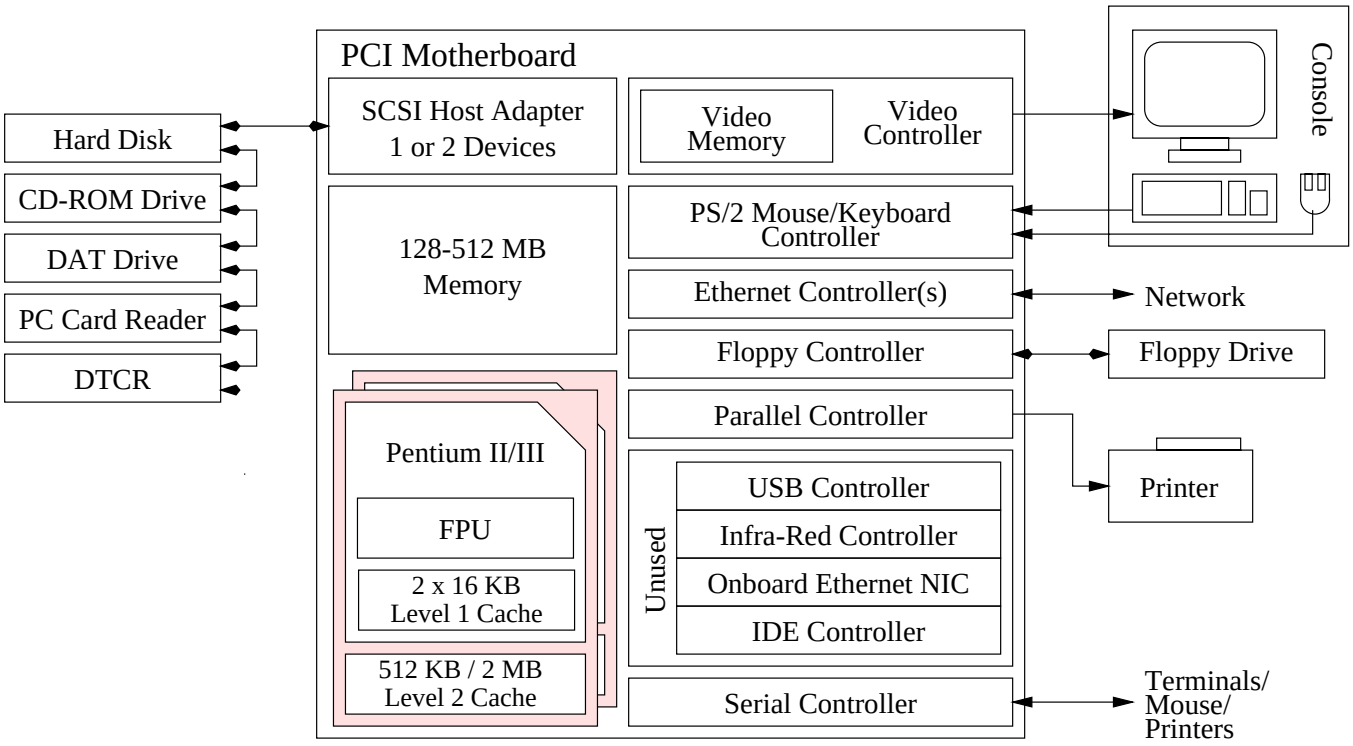

Figure 3.1. Intel 440 Motherboard Hardware Layout

- Mouse or Touchpad
- 4-Port Interface Controller Card (optional)
- 2/4 port Ethernet Card (optional)
- PCL5 Parallel Printer

A more detailed list of the COTS products used can be found in Appendix A of this document. Figure 3.1 shows the hardware layout of the XTS-300. Each major component is discussed in this chapter.

3.2 Central Processing Unit

The XTS-300 utilizes one or two Pentium II or Pentium III processors as its CPU. Both kinds of CPUs support 32-bit addressing. They incorporate the following capabilities: cache, memory management, floating point coprocessing, and write buffers. The internal cache has 16 Kbytes for instructions and another 16 Kbytes for data. Both CPUs use a paging and segmentation mechanism that can be independently enabled and disabled. They also provide a ring protection structure (described in Section 3.4.3, page 22).

The microcode in the Pentium II/III is downloadable. To perform a download, software must issue a write to a specific "model-specific register." Writing to model-specific registers is a privileged operation, so untrusted software can not do it. The TCB does not use this download capability, but the BIOS does. Each CPU must be independently loaded and must be loaded during every system reset. The CPU hardware will reject the download if its version (which is encoded within the download) is not acceptable or if the encoding is wrong. Intel strongly protects the syntax and semantics of the microcode and uses strong encoding techniques (which are also highly proprietary) to prevent malicious users from corrupting or substituting the microcode image within the BIOS.

3.2.1 Registers

Both CPUs include the following sets of registers (quantity in parentheses):

- General (8)
- Control (5)
- Instruction Pointer (1)
- EFLAGS (1)
- Segment (6)
- Memory Management (4)
- Floating Point (8)

The Intel Pentium II/III CPU also supports "model-specific" registers. Note that the microarchitecture also has many "hidden" registers that are not visible to the XTS-300.

3.2.1.1 General Registers

These 32-bit, general registers hold the data necessary for logical and arithmetic operations, as well as for address calculations. Each 32-bit register can be broken down into two 16-bit registers, where only the low 16 bits are used. Each 16-bit low register can be broken down even further into a high byte and a low byte. General registers are accessible by applications.

3.2.1.2 Control Registers

There are several 32-bit Control Registers: CR0 - CR4. CR0 contains system flags that control the state of the entire system rather than of any individual task. CR1 is reserved. CR2 and CR3 are used by the paging mechanism as follows: CR2 contains information to handle page faults, and CR3 contains the base register of the Page Directory (PDIR). CR4 contains additional system flags. Control registers are generally not accessible outside of Privilege Level 0 (PL0).[2]

CR0 contains system control flags that are general to the processor. The following flags are contained in CR0:

Paging Bit (PG) Enables or disables paging mode.

Cache Disable (CD) Enables and disables internal L1 caching.

Not Write-through (NW) Enables and disables cache write-throughs.

Alignment Mask (AM) Enables and disables alignment checking.

Write Protect (WP) Write-protects user-level pages against supervisor-level writes when set.

Numeric Error (NE) Enables and disables the mechanism for reporting floating point numeric errors.

Task Switched (TS) Set for every task switch. Used when executing floating-point arithmetic.

Emulation (EM) Enables and disables the math coprocessor.

Math Present (MP) Used to help synchronize processor with the coprocessor.

Protection Enable (PE) Enables and disables segment level protection.

The CR3 register contains two flags: Page-level Cache Disable (PCD) and Page-level Writes Transparent (PWT). The PCD controls caching of the referenced page (which will be a paging directory). This bit is set to disable caching by the kernel startup and during process creation. The PWT controls write-through of the referenced page on an external cache, but is irrelevant since caching of the referenced page is disabled. The CR3 register also contains the 20 most significant bits of the base address of the page directory (See Section 3.3.3.1, page 16 for a discussion of page directories).

The CR4 register contains flags to control the following features: machine check, 4 Mbyte page frames, debug extensions, virtual 8086 extensions, new time stamp instructions, 36 bit addressing and other miscellaneous features. None of these features is used by the TCB and none of them are accessible outside PL0.

[2] An exception is that part of CR0 can be read outside of PL0.

3.2.1.3 Instruction Pointer Registers

The instruction pointer register (EIP) contains the 32-bit pointer to the offset of the address in the current code segment of the next instruction to execute.

3.2.1.4 EFLAGS Register

The EFLAGS register handles virtual-8086 mode, I/O privilege, task switching status, debugging, and maskable interrupts. Only the security relevant bits, such as the I/O Privilege Level (IOPL) and the Interrupt-enable Flag (IF), are described in detail. The following is a list of bits in the EFLAGS register:

Carry Flag (CF)

Parity Flag (PF)

Auxiliary Carry Flag (AF)

Zero Flag (ZF)

Sign Flag (SF)

Trap Flag (TF)

Interrupt Enable Flag (IF)

Direction Flag (DF)

Overflow Flag (OF)

Input/Output Privilege Level (IOPL)

Nested Task (NT)

Resume Flag (RF)

Virtual 8086 Mode (VM)

Alignment Check (AC)

CPU ID - used in conjunction with the CPUID instruction

Virtual 8086 interrupt pending

Virtual 8086 interrupt flag

The IOPL bits determine if a process has access to an I/O address space. If the IOPL is greater than the current privilege level, access is granted. Otherwise, unless the Task State Segment (TSKSS[3]) bitmap allows access, an exception is incurred.[4] In the XTS-300, the IOPL field is set to 0.

[3] Intel refers to this as the TSS; however, because TSS also represents the TCB System Services on the XTS-300, a different acronym was needed.

[4] See Section 3.4.1, page 21 for a complete description of the TSKSS.

The IF bit, if set, will enable the processor to respond to maskable interrupts. Maskable interrupts are interrupts in an executing program caused by such things as printing. A maskable interrupt, as opposed to a non-maskable interrupt (NMI), will complete the current task before servicing the interrupt.

The IF, IOPL, RF, VM, and AC flags cannot be modified by processes running at a privilege level not equal to 0. These flags are considered privileged on the CPU. The virtual 8086 interrupt flags supported by the Intel Pentium II/III are also privileged, but are ignored because virtual 8086 mode extensions are disabled by bit settings in CR4.

3.2.1.5 Segment Registers

A segment is an independent, protected address space. Each segment register is designated to store information about a certain type of segment. The CS register, for example, refers to a code segment that contains instructions. There are six 16-bit segment registers on the CPU; Code Segment (CS), Data Segment (DS), Stack Segment (SS), Extended Data Segment (ES), Far Data Segment (FS), and Global Data Segment (GS). When an instruction is executed, it loads a pointer to a descriptor table into the segment register. This pointer is called the segment selector. In addition to holding a pointer to a descriptor table, the selector also contains information about the type of descriptor table being accessed and privileged information. The CPU allows a task to have access to as many as six segments at one time. Section 3.3.3.1 describes segments in greater detail.

The SS register refers to information regarding the current stack segment. The segment being pointed to by the SS register must have a privilege level which is equal to or greater than that of the process in order to read or write. The SS register can be read and written by processes running at all privilege levels.

The DS, ES, FS, and GS registers refer to segments containing data. There are four data segments, to improve performance. The segment being accessed is always checked to ensure that the proper registers reference the appropriate corresponding segment (i.e., data segments are only referenced by data segment registers and code segments are referenced only by code segment registers). See Section 3.4.3.1, page 23 for more discussion on how and when these checks are performed.

3.2.1.6 Memory Management Registers

The CPU contains four memory management registers; the Global Descriptor Table Register (GDTR), the Local Descriptor Table Register (LDTR), the Interrupt Descriptor Table Register (IDTR) and the Task Register (TR). These registers are initially set by the System Loader and are modified only by Ring 0 operations thereafter.

The GDTR contains the 32-bit base address and the 16-bit segment limit of the Global Descriptor Table (GDT). The GDT contains a list of segment descriptors that are used globally. In other words, the GDT contains information which must be accessible to the system at all times. The LDTR contains the 32-bit base address and the 16-bit segment limit of the Local Descriptor Table (LDT). The LDT contains a list of segment descriptors that are local to the current process. The IDTR contains the 32-bit base address and 16-bit segment limit of the Interrupt Descriptor Table (IDT). The IDT contains a list of gate descriptors associated with the handling of interrupts. The TR register contains the segment selector for the current task. This selector points to the TSKSS contained in the GDT for the current process.

3.2.1.7 Floating Point Registers

There are eight 80-bit floating point registers in the FPU on the CPU. These registers handle the floating point calculations. The values within these registers are explicitly saved and restored by the operating system for each process. However, they are not saved when a task switch occurs.

To keep track of the general state of the Floating Point Unit (FPU), a 16-bit status word is used. This FPU status word contains flags which can be passed back to the CPU through the AX register. This status word includes a stack fault flag and six exception flags, including overflow and underflow flags. If a numeric exception occurs, the FPU will either handle the exception internally (i.e., it will produce the most reasonable result based on the information and allow the execution of the process to continue un-interrupted) or by sending either an interrupt 16 (Floating Point Exception) or an external interrupt to invoke a software exception handler. The XTS-300 is configured to handle these exceptions by setting the NE bit in the CR0 register.

3.2.1.8 Intel Pentium II/III Model-Specific Registers

The model-specific registers are not used by the TCB and are treated as privileged by the hardware, i.e., they are not accessible outside PL0. The model-specific registers include: machine check address, machine check type, time stamp counter, event control, and event counters. Note that the XTS-300 sets a bit[5] to disable outer ring access to the time stamp counter.

3.3 Memory on the XTS-300

The CPU supports several types of memory. The following types of memory are supported by the XTS-300:[6]

- Main Physical Memory (up to 64 Gigabytes)
- Cache Memory

3.3.1 Main Physical Memory

The main physical addressing range for the CPU is 64 Gbytes. The XTS-300 can use up to 862 Mbytes. Figure 3.2 shows the physical address space on the XTS-300.

Proprietary figure removed

Figure 3.2. XTS-300 Use of Physical Address Space

Once the bootstrap loader has been executed, physical memory locations are rarely referenced by software using a physical address.[7] Instead, a virtual address is referenced. The virtual address is then mapped, via a linear address, to the main physical address.

[5] Specifically the TSD bit in the processor's CR4 register

[6] The XTS-300 supports other types of memory including video memory, CMOS/NVRAM, BIOS ROM and device memory mapped I/O addresses. However, they are not covered in this section.

[7] The Kernel declared data segment is setup so that the linear address is identical to the physical address.

3.3.2 Cache Memory

The Intel Pentium II/III contains separate 16 Kbyte write-back data and 16 Kbyte write-back instruction L1 caches. The Intel Pentium II/III caches use a "write-back" policy in which main memory will only be updated when necessary (either when lines need to be reused in the cache or an external bus master needs to access the same physical memory locations). The internal L2 cache (up to 2 Mbytes) is enabled (via BIOS).

Caching is used on the XTS-300 to increase performance. The caches are not directly addressable by a process.

The XTS-300 disables the caching of PDIR entries because these entries are cached in the Translation Lookaside Buffer (TLB). Caching never takes place at both the PDIR and Page Table levels. The CD and NW bits in CR0 are both set to zero, enabling the internal L1 cache write-backs. The TCB never uses the PWT bit. The cache is never disabled after the TCB begins execution.

Upon startup, the caches are cleared. When a pagetable changes (due to segment growth), the paging cache will be flushed to be sure that the new pagetable entries and PDIR entries come into effect. Flushing the paging cache consists of saving and reloading the CR3 (i.e., PDIR pointer) registers. Segment-level cache is flushed by loading a NULL selector into the volatile segment registers (see Section 3.3.3.1 and Section 3.3.3.2 for more information on segments and pages).

3.3.3 Virtual Memory

The XTS-300 uses a virtual memory management scheme to address physical memory locations. The XTS-300 processor's virtual memory management scheme supports a 64 Tbyte non-process virtual memory address range. Each process is allowed up to 4 Gbytes of virtual memory space.

3.3.3.1 Segments

A segment is a unit of address space ranging from 4 Kbytes up to 4 Gbytes, though the operating system limits segments created by a process to 4 Mbytes, and is defined by the operating system. Segments are defined by a segment descriptor. The segment descriptor contains information about the size, base address, and protection level of the segment (as seen in Figure 3.3).

A segment register holds a selector that points to the offset within the descriptor table where a particular segment descriptor resides (see Figure 3.3). The purpose of the segment register is to provide the initial information required to locate a page in memory; the logical address. The type of segment register that is used (e.g., code segment, data segment, or stack segment) is dependent on what type of segment is being accessed. For example, if instructions are being accessed, the CS register will be used. The segment register consists of two parts; a visible part (loaded using the MOV instruction) that contains a segment selector, and an invisible part (loaded by the processor and invisible to even the kernel software) that contains information about the limit, the base address, etc. The invisible parts of the segment registers constitute a segment cache. The segment selector contains an index to a specific segment descriptor in the descriptor table, a bit (TI) to identify which descriptor table is being accessed (GDT or LDT), and a requester privilege level (RPL) field that identifies the privilege level of the procedure that created the selector. On the XTS-300, segment descriptors are loaded into the GDT only during startup (though task descriptors are updated in the GDT each time a process is created). The GDT is also loaded with pointers to the LDT and gate descriptors.

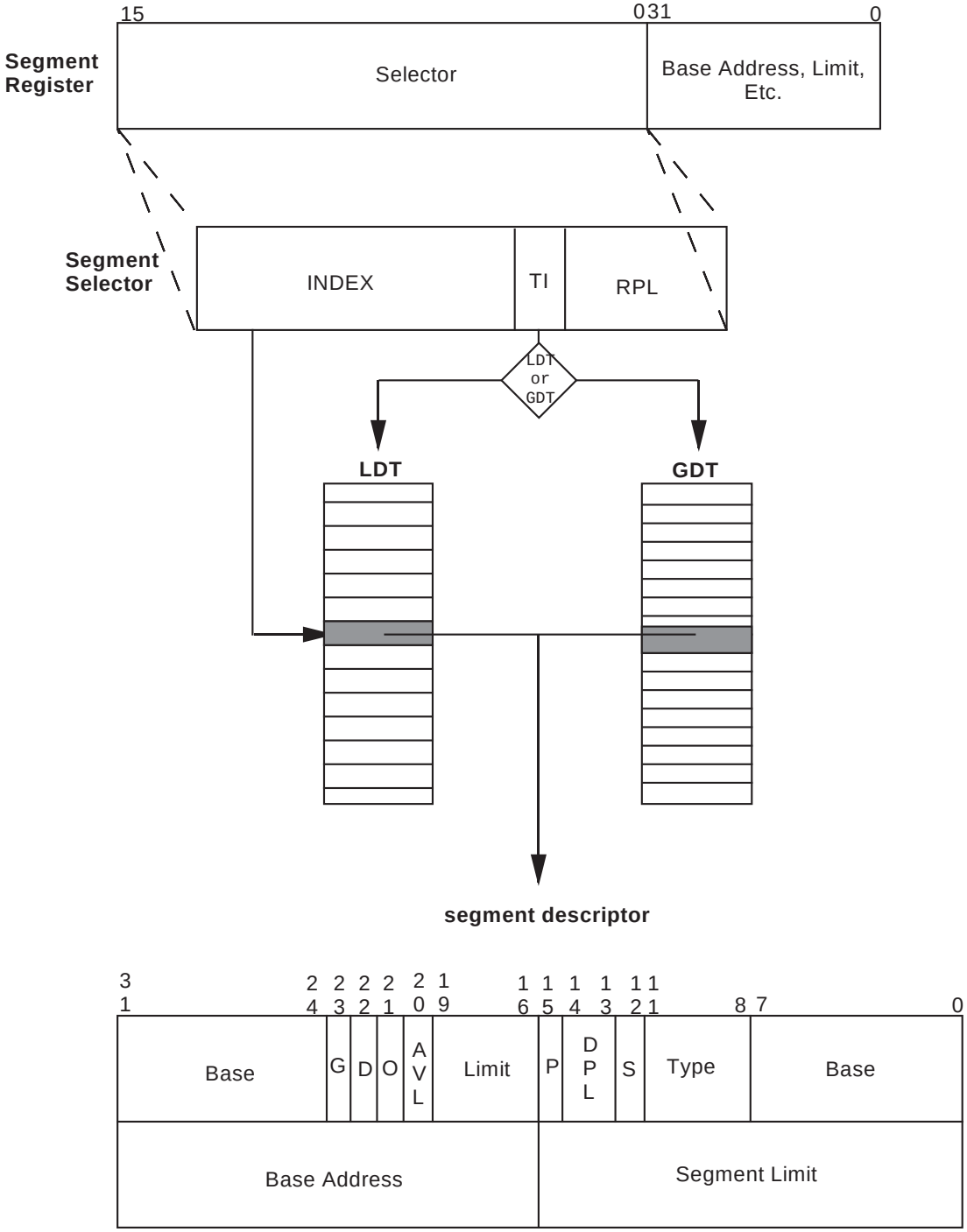

Figure 3.3. Selector-to-segment translation

However, the majority of segments are process local and only use the LDT. Figure 3.3 displays a graphical representation of the segment selector, the segment descriptor and the relationship between the two.

A segment descriptor provides information about the size (segment limit), base address, descriptor privilege level (DPL), and other control and status information. There are two classifications of segment descriptors, code/data descriptors and system (task and gate) descriptors. The classification of the descriptor is determined by the S bit (bit 12) in the descriptor itself. The maximum size of a segment can vary from 1 Mbyte to 4 Gbytes. The granularity bit (G) (bit 23) toggles the limit field of the descriptor to be interpreted as either bytes or pages. Segment descriptors are created by Ring 0 software and cannot by modified directly by an application. On the other hand, segment registers can be modified by applications.

3.3.3.2 Pages

A page is a 4 Kbyte portion of memory. A major distinction between segments and pages is that pages have a fixed size while segment size can vary. Unlike segmentation, paging is transparent to applications (except that Ring 2 can see paging violations due to an attempted write to a read-only page). On the XTS-300, segmentation and paging are combined such that segments are paged. A segment is capable of being 1024 pages. By using paged segments, an application does not have to reside entirely in main memory. Only the portion of the application which is actually being used (that is, the page being used) needs to be present in physical memory.

Figure 3.4 shows how a logical address, or virtual address, is translated into a physical address. Pagetables are intermediaries between the linear address and the physical address. The linear address, obtained from the segment, contains three pieces of information about a page; the PDIR entry location, the pagetable entry location, and the location in the page frame.

The Page Directory (PDIR) is a table of pointers that point to specific pagetables associated with a particular process. On the XTS-300, each process has its own PDIR. CR3 contains the information about which PDIR should be accessed. The PDIR can contain up to 1024 pointer entries, each pointing to a particular pagetable associated with its process. With the exception of Binary Compatibility Standard (BCS) (see Section 4.2, page 41), kernel text and declared data segments and the Global Pool segments, most segments use only one, unique PDIR entry number. This number is the same as the segment number. Each location in a pagetable is a pagetable entry. The pagetable entry is located by using the information in the linear address. The pagetable entry points to a particular page frame. Information in the linear address specifies where in the page frame to look.

3.3.3.3 Address Translation

Figure 3.4 depicts the sequence of events that occur during address translation. At a high level, the processor begins with a virtual address. With that virtual address, a linear address is determined. The information contained in the linear address, combined with the proper PDIR and pagetable entries, define where in a page frame the physical address is located.

The address translation begins with a selector and offset. The selector is obtained either from a 48-bit pointer or from a segment register. The selector indexes into the LDT or GDT to a segment descriptor. The descriptor defines the linear base address of the segment. The base address field in the segment descriptor, along with the original offset, form the linear address.

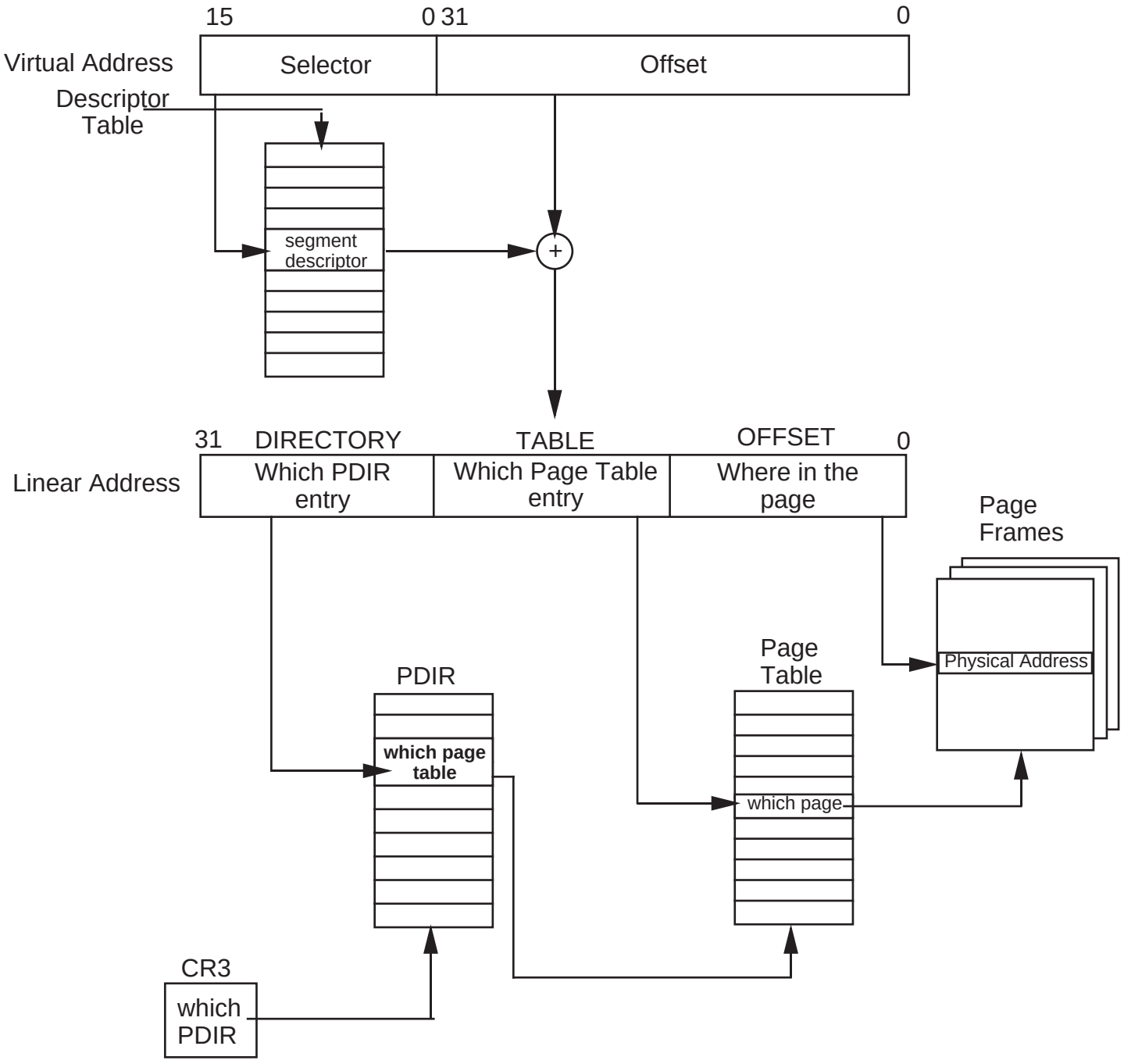

Figure 3.4. Logical-to-physical (or page) translation

The linear address contains three pieces of information; the offset within the PDIR, the offset within the pagetable, and the offset within the page frame. CR3 points to the specific PDIR associated with the current process. Once a PDIR entry has been located, its contents are used to determine the location of the pagetable entry. The PDIR entry points to the specific pagetable. The pagetable entry points to a particular page frame. This information, along with the offset field in the linear address, is used to determine where in the page frame the physical address is located. A description of how the ring protection mechanism is used during page translation is contained in Section 3.4.3, page 22.

3.3.3.4 Gates

In order to protect control transfers between code segments of different privilege levels, gate descriptors are used. There are four types of gate descriptors; call gates, trap gates, interrupt gates and task gates.

Interrupt and trap gates are used to reference procedures and tasks. When an interrupt occurs, the IDTR register and interrupt number are used to locate the interrupt or trap gate in the Interrupt Descriptor Table (IDT). This gate contains a pointer to either the GDT or the LDT offset containing a segment descriptor. The segment descriptor and the gate combined yield the interrupt procedure entry point. The gate descriptor contains the offset within the segment of the interrupt procedure entry point.

A call gate handles the control transfer from one segment to another within the same task. This is done by issuing either a JMP or a CALL instruction. Call gates may reside in either the GDT or the LDT, but never in the IDT. Call gates contain the segment selector that points to a gate descriptor that, in turn, points to an offset to the procedure entry point. Because the segment selector points to the gate descriptor in a call gate, there is no use of the offset field in the segment selector. Instead, the offset of the gate descriptor is used along with the base address field in the code segment descriptor to create a procedure entry point.

The task gate is utilized when a task switch is implemented. The task register (TR) contains the index of a task descriptor in either the LDT or the GDT, but the XTS-300 keeps all task descriptors in the GDT. This descriptor defines a particular TSKSS, which is explained in further detail in Section 3.4.1.

A gate descriptor allows for a transfer of control from one segment privilege level to another. Gate descriptors contain information about the destination segment. A description of how protection is enforced during control transfers and task switching is discussed in Section 3.4.3.

3.3.3.5 Returning from Gates and Interrupts

Once a control transfer has been completed or an interrupt has been serviced, a transfer back to the original process is required. This is performed using the IRET and RET instructions.

The RET instruction is used to return from a CALL instruction. When used, a check is made of the descriptor addressed by the return. The code segment to return to must be of equal or lesser privilege than the current code segment. A return is granted if the RPL of the code segment is numerically greater than the CPL of the segment being returned from and if the DPL of the code descriptor is greater than the CPL. If the CPL is less than the return code segment DPL, the segment registers are loaded with a NULL selector (see Section 3.4.3 for a complete definition of RPL, CPL, and DPL).

The IRET instruction is used to return from an interrupt. The privilege checks made on an IRET instruction are dependent on the value of the NT flag in the EFLAGS register. If NT is equal to 0, the IRET instruction

returns from the interrupt without performing a task switch. A check is performed to ensure that the code returned to is equally or less privileged than the interrupt routine. If the NT flag is equal to 1, the IRET instruction will reverse the operations previously performed by the CALL/INT instruction.

3.4 Process Management

A task is a program which is currently being executed. The CPU uses privilege level checking to distinguish between the different privilege levels within each process. The privilege level checking ensures that no access is given to more privileged segments and instructions from less privileged code within the process. The following three sections describe in greater detail: processes, interrupts and exceptions, and hardware protection devices.

3.4.1 Processes

A process is instantiated through either an interrupt, an exception, a JMP, or a CALL. A context switch is the act of one process being saved in order for another process to execute. Context switching occurs in any of the following four cases:

- The current task executes a JMP or CALL to a TSKSS descriptor
- The current task executes a JMP or CALL to a task gate
- An interrupt or exception indexes to a task gate in the IDT
- The current task executes an IRET when the NT flag is set

When a process is saved in memory, it is saved to a TSKSS. The TSKSS contains information about the general registers, the segment registers, the EFLAGS register, the instruction pointer, the selector for the TSKSS of the previous task, the selector for the LDT, the PDIR pointer (CR3 value) and the I/O bit map.[8] The Intel Pentium II/III CPU also allows an "interrupt redirection map" to reside in an extended area of the TSKSS, but this is only valid in virtual-8086 mode and is never used by the TCB.[9] A descriptor for the TSKSS is stored in the GDT.

The current task is identified by the TR. The TR points to the TSKSS of the current task. When a task is switched, the old TR is stored and the new TSKSS is loaded. If that task is later suspended, the instruction pointer for that task is saved so that the task can later be resumed at the same point at which it was suspended.

When a task is switched, the TS bit in the CR0 register is also set. This bit provides coordination between different tasks and the Floating Point Unit (FPU). The TS bit signifies that, depending on whether it is set or not, the context of the FPU may not be that of the current task. The TS bit is always set to 1 by the CPU when a context switch occurs. It is only reset to zero by software if an FPU instruction is executed by the currently executing process. This bit is readable by an untrusted process.

[8] The TSKSS is always a fixed size and includes enough storage for the I/O bitmap to include the video controller ports.

[9] On the XTS-300, the I/O bit map is set to disable access to all I/O ports when a process is created. However a process can be granted exclusive access to the video controller ports, in which case certain bits in the I/O bitmap of that process will be toggled on and off as needed, by the TCB.

3.4.2 Interrupts and Exceptions

The CPU supports both interrupts and exceptions. Interrupts occur when there is a forced program control transfer due to hardware. An exception is caused when either an instruction cannot be performed or when the INT instruction is executed. There are three types of exceptions: faults, traps and aborts. In the case of a fault, the saved contents of the CS and EIP registers point to the instruction before the instruction that generated the fault. In the case of a trap, however, the exception is generated after the faulting instruction is performed. Aborts are used for major errors such as hardware errors or inconsistent values in system tables. The CPU handles all software level INT calls as exceptions, though interrupt gates may be used to process the calls. The XTS-300 sets the privilege level in the interrupt and trap descriptors in the IDT such that only interrupt 3 (breakpoint) and 4 (overflow) can be called from outside PL0.

The CPU contains 256 interrupt address spaces. The first 32 locations are reserved for exceptions and NMIs. The XTS-300 uses 16 additional interrupt locations for its own interrupts.[10] Each interrupt code has an associated gate with a corresponding gate descriptor. These gate descriptors are stored in the IDT. All interrupts, excluding NMIs, have the same priority level on the XTS-300.

Interrupts on the Intel Pentium II/III base, are controlled by the advanced programmable interrupt controller (APIC). An APIC is integrated into the Intel Pentium II/III CPU. The different APICs cooperate and are designed to support multiprocessing. The older legacy I/O APIC functionality, present for compatiblity, is not used by the XTS-300.

If a less privileged process produces an interrupt with a higher numbered privilege, a check is made on the CPL associated with the process and the DPL of the interrupt code segment. If the CPL is less than the DPL (i.e., the process has a lesser privilege than that of the interrupt routine), a stack switch occurs. Because servicing the interrupt, while in untrusted code, requires a change in privilege levels, a switch will be made to the Kernel stack. The reason for this is to prevent less privileged programs from manipulating more privileged programs. The contents of the registers are saved in the new stack and the registers are loaded with the new, more privileged, stack information. When the interrupt has been serviced, the original contents of the registers are restored and the process continues. This prevents a lesser privileged process from writing to a more privileged process.

The XTS-300 is configured such that all SCSI devices use the same interrupt request lines (IRQs). Ethernet lines can share the same IRQ. Each communications port (i.e., COM1 and COM2) has its own IRQ and all SIO4 ports (see Section 3.7.1, page 30) have the same IRQ.

3.4.3 Protection

The Intel Pentium II/III has the ability to operate in four different modes; protected mode, real-address mode, and virtual 8086 mode, and system management mode. The XTS-300 runs in real-address mode at start-up and then switches to protected mode. Virtual 8086 and system management modes are not utilized by the XTS-300.

Protected mode utilizes most of the instructions and architectural features of the CPU, including all of the protection features. It uses the 32-bit instruction set. Real-address mode simulates an enhanced 8086 processor.

[10] This does not take into consideration the APIC specific interrupts.

A ring architecture, provided by the CPU, is used to restrict access to segments, pages, and instructions. The CPU ring architecture consists of four levels (PL0-PL3), with PL0 being the most privileged level. As seen in Figure 3.5, a non-TCB process utilizes a different ring architecture than a TCB process. Both, however, share the same Ring 0 and Ring 1 assignments. These four ring levels are used at several different levels of address translation. Bits are used to supply privilege information about the selector (RPL), descriptor (DPL), and process levels (CPL).

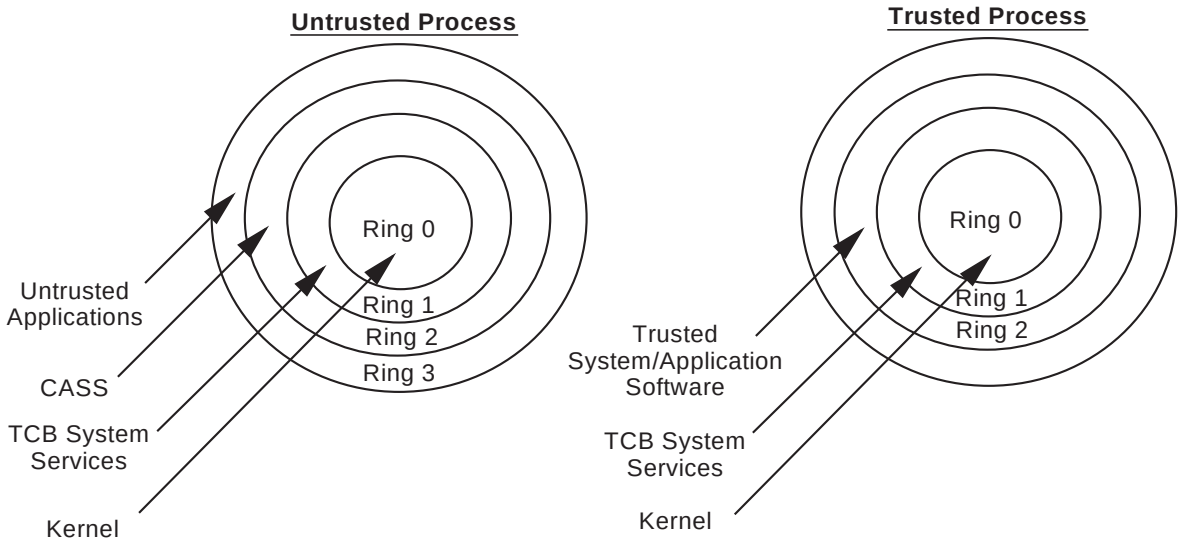

Figure 3.5. Privilege Level

The CPU provides five ways by which it checks for protection violations contained in memory references; type check, limit check, restriction of addressable domain, restriction of procedure entry points, and restriction of instruction sets. If one of these checks does not pass, an exception is generated.

3.4.3.1 Type Checking

Type checking is a test of the Type field in the segment descriptor to ensure that code segments are treated as code segments and data segments are treated as data segments. Type checking is performed on segments. Because segments can contain either application code and data segments or system segments and gates, protection is necessary at the segment level to ensure that all segments are properly defined and that no segment is accessed by a less privileged process. There are two cases when the CPU will perform a type check on a segment. The first check is made when a descriptor selector is loaded into a segment register. This is to ensure that the CS register, for example, contains only code segment selectors. The second case is when a segment is used by an instruction. This is because of three architectural rules that state the following:

- Executable segments cannot be written to by an instruction
- A data segment must have its writable bit set in order for an instruction to be able to write to it
- An executable segment must have its read bit set for an instruction to read it

3.4.3.2 Limit Checking

Limit checking ensures that programs do not try to access addresses that are located outside of the targeted segment. Limit checking also occurs at the segment level. The limit field in the segment descriptor defines its size. The limit field is dependent on the G bit that is also in the segment descriptor. The granularity bit toggles the maximum limit between 1 Mbytes and 4 Gbytes. The limit field is also dependent on the expand-down bit; however, this is not utilized on the XTS-300. A general protection exception is generated if access to a memory byte, word, or doubleword at an address greater than the limit is attempted.

3.4.3.3 Restriction of Addressable Domain

Restriction of addressable domain refers to protection checking at the paging level. When a reference is made to a page in memory, the processor checks the CPL against the User/Supervisor (U/S) bit of the pagetable entry. If the U/S bit is set to 0, then that page is set to supervisor level. If the U/S bit is set to 1, then the page is set to user level. If the CPL is 0,1, or 2, then the processor will be running at the supervisor level and all pages will be accessible. If the CPL is 3, the processor will be running at the user level and only the pages marked U/S bit = 1 will be accessible. The XTS-300 sets the U/S bit to 1 which means that all pages are readable and writable by any level.

In addition to the U/S bit, there is a Read/Write access (R/W) bit at the page level. If the processor is running at supervisor level and the Write Protect (WP) bit in the CR0 register is set to 0, then all pages are both readable and writable. However, if the processor is running at user level, then only the pages with R/W = 1 will be readable and writable. The XTS-300 always sets WP to zero and only sets R/W to 0 for pages in Ring 3, read-only shared memory segments. Emphasis is, in turn, placed on protection at the segment level.

3.4.3.4 Restriction of Procedure Entry Points

Restriction of procedure entry points relies on the call gate mechanism. When a CALL or JMP instruction is executed, a call gate is employed to handle the transfer from one code segment to another. The call gate specifies where in the procedure to enter (i.e., what segment) and what privilege levels are allowed access to that segment. When a control transfer is made using a call gate, four different privilege levels are checked against each other; the DPL of the call gate, the DPL of the code segment descriptor, the CPL of the current code segment selector, and the RPL of the call gate selector.

The following describes the sequence of events for privilege level checking when a call gate is executed:

The CPL, located in the code segment register, is compared with the RPL of the gate selector (located in the call instruction). The maximum numerical value of the two is taken to be the active privilege level. This calculated privilege level is then compared to the DPL of the gate descriptor. The maximum of the RPL and CPL must be numerically less than or equal to the DPL in order to continue; otherwise, a general protection trap is produced. If this check passes, a comparison is made of the DPL of the destination code segment and the active privilege level. If the active privilege is not numerically greater than or equal to this second DPL, a general protection trap occurs and the call will not be allowed to proceed.

If a gate is not used, the sequence of events differs in one area. The last check between the CPL and the DPL of the destination code segment requires that the DPL be numerically less than or equal to the CPL.

3.4.3.5 Privilege Level Checking

The CPU uses the three privilege indices, mentioned at the beginning of this section, to make decisions on the accessibility of data and code. The CPL, RPL, and DPL provide a mechanism for isolating privilege levels and preventing any mixing of levels. A privilege level check is made at various points during address translation to ensure that no mixture of privileged data takes place. This section covers privilege level checks made when address translation is made for data access only.

The first check to be made during address translation is when the selector is loaded into the segment register. The selector contains the RPL of the process. The descriptor that the selector points to has a DPL associated with it. The RPL and the DPL are compared to ensure that the RPL is numerically greater than or equal to that of the DPL. If this is not true, a general protection exception will be issued. All privilege checks are made at the segment level, except as discussed in the section above: Restriction of Addressable Domain.

3.4.3.6 Ring 0 Privileged Instructions

The following instructions are privileged on the Intel Pentium II/III and can only be executed at PL0:

- CLTS -Clear Task-Switched Flag
- HLT -Halt Processor
- INVLPG -Invalidate TLB Entry
- LGDT -Load GDT Register
- LIDT -Load IDT Register
- LLDT -Load LDT Register
- LMSW -Load Machine Status Word (modifies the lower 16 bits of CR0)
- LTR -Load Task Register
- MOV to/from CRn -Move to Control Register 0, 2, or 3
- RDMSR -Read from model-specific register
- WRMSR -Write to model-specific register
- RSM -Return from system management mode
- INVD -Invalidate cache
- WBINVD -Write-back and invalidate cache

The RDTSC instruction is optionally privileged, depending on the setting of a bit in CR4. XTS-300 configures CR4 so that RDTSC is privileged.

3.5 Input/Output

Input and output (I/O) on the XTS-300 is accomplished through I/O ports that are actually registers connected to peripheral devices. The I/O ports are separate from both the Basic Input/Output System (BIOS), which acts as a liaison between the system software and the hardware during system bootstrap, and the main physical memory. The I/O ports are also separate from the "configuration space" implemented by the PCI chipset on the Intel Pentium II/III motherboard.

The CPU provides a separate protection mechanism for I/O called the IOPL. The IOPL, located in the EFLAGS register, determines the lowest privilege level at which access to the I/O ports is possible. Due to the fact that each process has its own copy of the EFLAGS register, there can exist a different IOPL for every process. The XTS-300 sets the IOPL field to be zero, leaving only PL0 access. A process can change the IOPL using the POPF instruction; however, this change can only be made while in Ring 0.

Untrusted software can access I/O ports only through TCB gates (at the software level), except that the TCB can grant exclusive access by a process to the video controller ports. This provides protection against access to the I/O ports by software.

The TSKSS also provides an I/O permission bitmap that contains information about I/O addresses to which a particular process has access even if the CPL of the process is greater than the IOPL. However, this use of the bit map is disabled during process initialization by the XTS-300. A process which has read/write access to the console device can gain direct access to the video controller registers via a TCB request. If access is granted by the TCB, the TCB enables particular bits in the I/O bitmap of the process which correspond to the video controller ports. The TCB disables those bits if a secure attention event occurs.

The following instructions are "sensitive instructions" (they will only execute if the IOPL field is greater than or equal to CPL), so on the XTS-300 they can only be used when operating at Ring 0:

- IN -input
- INS -input String
- OUT -output
- OUTS -output String
- CLI -clear Interrupt-Enable Flag
- STI -set Interrupt-Enable Flag

3.6 Primary XTS-300 Motherboard Chips

The motherboard is the heart of the XTS-300 processing hardware. Depending in the XTS-300 model, the motherboard will come in one of two flavors. The Model 300 uses the Intel LX motherboard while the Model 500 uses the GX motherboard. The GX is a minor evolutionary step beyond the LX and is extremely similar in functionality and layout. Where apppriate, the differences will be documented.

Note that only a single motherboard is used, there are no daughter boards which extend the central buses, and all CPUs reside on the motherboard. The motherboard hosts a number of relatively complex subcomponents,

including the CPU, hosts several controllers into which peripheral devices can be directly connected, and hosts the "internal" buses into which all additional controllers are connected.

The Intel Pentium II/III motherboard has the following features:

- Intel Pentium II/III CPU support with L1 and L2 internal caches
- Designed for and supports multiprocessing
- Contains a PCI bus, ISA bus, and APIC bus as well as a host bus, maintenance bus and X bus.
- Contains a complex interrupt subsystem, including APIC and SMI support (though SMI is disabled).
- 64-bit data path to main memory
- Support for upto 64 Gbytes of main memory
- ISA devices and the keyboard controller are packaged as highly integrated chips
- Some additional controllers are integrated onto the motherboard: COM1/COM2, parallel, video, SCSI host adapter, IDE (which is not used), and mouse.
- Version of BIOS which is software upgradable (partially) and which includes SCSI setup

In addition to the CPU, the Intel Pentium II/III motherboard contains the following major chips:

- Intel 82443LX (LX) or 82443GX (GX) PCI/AGP Controller (PAC)
- Intel 82371AB (LX) or 82371EB (GX) PCI-ISA Bridge (PIIX4)
- National Semiconductor PC87307 (LX) or PC87309 (GX) Super I/O
- Adaptec AIC-7880 (LX) or AIC-7896 (GX) SCSI Host Adapter
- Cirrus Logic CL-GD5446 (LX) or CL-GD5480 Video Controller

The video controller and SCSI controller are discussed in the "Peripheral Controllers and Devices" section below. The on-board IDE controller, SMM and advanced power management, second floppy controller, USB port, infra-red, Intel NIC chip and Power management features are not used by the XTS-300.

The chips discussed below can only be programmed by software through specified I/O ports (except that APIC subcomponents use memory-mapped locations, which are not accessible outside the Kernel). Due to IOPL and I/O bitmap settings established by the TCB, none of these ports are accessible from outside PL0.

3.6.1 PCI AGP Controller PAC

The PAC chip provides the following major functions: memory control, host-to-PCI bridge, and AGP interface. These functions appear as two separate PCI devices, with the memory control and bridge in the first PCI device. AGP is not used on the XTS-300. The PAC controls access to system memory by the CPU, PCI and ISA bus masters. It mediates all requests from the CPU to PCI and ISA devices, part of which is

to translate the 64-bit, 66MHz (or 133 MHz for GX) host bus protocol to the 32-bit, 33MHz/66MHz PCI bus protocol, and vice versa. It also performs bus arbitration for the PCI bus.

The PAC provides buffering to avoid bus wait states as much as possible. Note that though the CPU allows 36-bit addresses, the PAC only allows 32-bit addresses.[11] The GX PAC is an evolutionary step beyond the LX PAC, but is extremely similar. The GX PAC boots the host bus speed from 66MHz to 133MHz, supports a larger theoretical main memory size, and supports additional power management features.

3.6.2 PCI-ISA Bridge (PIIX4)

The PIIX4 appears as four logical PCI devices: IDE unit, USB unit, advanced power management unit, and the PCI-ISA bridge. Only the last of these is used on the XTS. The PIIX4 provides address, data, and control line paths (and translation) between the PCI and ISA buses. It also contains data buffering in both directions to boost performance. The PIIX4 allows concurrent PCI and ISA activity. The PIIX4 also directly implements a number of familiar ISA functions: bus control and arbitration, DMA controller, master and slave PICs, PIT timers, X bus bridging, and RTC.

3.6.2.1 DMA Controller

The DMA controller function supported incorporates two 8237-compatible DMA controllers attached as master and slave. Channel 4 is used for the cascade connection, so a total of seven channels is available. The only DMA channel used is for the floppy controller (#2).

3.6.2.2 Programmable Interrupt Controller PIC

The PIC(s) functions supported on the PIIX4 incorporates two 8259-compatible PICs attached as master and slave. IRQ 2 is used for the cascade connection, leaving 15 IRQs that can be generated. IRQ 0 is hard-wired to the PIT and IRQ 8 is hard-wired to the RTC.

3.6.2.3 Programmer Interval Timer PIT

The PIT supported on the PIIX4 incorporates a single 8254-compatible timer unit. The unit supports three timers (0–3). As on the previous motherboard, timer 0 generates IRQ 0 and is used by the Kernel for time-of-day and for unique identifier generation. Timer 2 is used by the TCB to produce varying speaker tones (this was also true on the previous motherboard). Timer 1 is used to implement refresh, which occurs without OS and BIOS intervention.

3.6.2.4 Real Time Clock

The real-time clock is backward compatible with the real-time clock unit on the previous motherboard. The RTC is bundled with 256 bytes of battery-backed RAM (compared to 50 bytes of ISA CMOS on the

[11] The motherboards have limited main memory to 2 Gbytes while the XTS-300 only uses 862 Mbytes maximum.

previously evaluated system). The RTC uses IRQ 8. The Kernel uses the RTC in periodic interrupt mode to implement alarms and delays.

3.6.3 Super I/O (SIO)

The SIO integrates a number of familiar ISA controllers: floppy, serial (COM1 and COM2), parallel, and keyboard/mouse. The SIO provides a simple 8-bit interface to these controllers. The SIO also contains advanced power management and Plug-and-Play features, but these are disabled by BIOS setup and are not used by the XTS-300. All the controllers can operate independently, except that they share the ISA bus interface logic.

3.6.3.1 Serial I/O

The SIO contains two identical 16550-type UARTs, though one of them has advanced features (which are not used on XTS-300) that can be used for extended UART mode and for infrared links. The UART FIFOs are not used by the TCB. Connection to external serial devices is achieved via RS232 protocol and handshaking selection. The SIO utilizes the universal asynchronous receiver and transmitter technology (UART). Each serial channel has its own UART and each UART defines a separate I/O port address space for communicating with the XTS-300. Each I/O port address corresponds with a particular UART register.

3.6.3.2 Floppy Controller

In addition, the SIO contains a floppy controller which is very similar to the 82077A. The floppy controller controls a 3.5 inch microfloppy disk drive. The drive can read and write either standard (1 Mbytes) or high (2 Mbytes) density disks. The XTS-300 uses only high density disks. The floppy disk can contain data at various MAC labels (see Section 6.2.1.1, page 86); however, data flow is controlled by the operating system. This ensures that the correct address is being accessed at all times. The floppy drive always passes information via the floppy controller and there is never direct communication between the floppy drive and any other component.

The floppy controller sits on a 24-bit ISA bus. Because of this, the floppy DMA buffers must reside in the low 16 Mbytes of main memory. Also the 16-byte FIFO in the floppy controller is enabled by software.

3.6.3.3 Keyboard Controller (KC)

The keyboard controller is PS/2 and compatible with older 8042A chips. The keyboard controller is actually a microcoded hardware module and as such, the module contains its own embedded ROM, RAM, and processor. The keyboard controller is connected to the keyboard by a serial line.

DMA is not used by the KC on the XTS-300 and RAM, which is not accessed by the system, is used by the BIOS. All data transfers and keyboard/KC commands are sent via I/O port addresses, which are inaccessible outside PL0.

The Secure Attention Key (SAK), which is used to invoke the trusted path function of the TCB, is provided by the keyboard. The SAK allows the user to enter the trusted mode of operation or to abort a process

within the trusted mode. Detection of the SAK key being depressed is handled by the operating system through interrupts. The SAK key for the XTS-300, while at the console, is the "system request" key.

The hardware does not interpret the SAK key. The keyboard generates IRQ 1. Commands can be sent to the keyboard, and the keyboard must acknowledge commands. The TCB sends commands to the keyboard to set the scan code set and to update the LEDs, and neither of these can be done directly by untrusted software.

3.6.3.4 Mouse

A PS2 mouse is standard on the XTS-300. The keyboard controller handles the PS/2 mouse. PS/2 mouse interrupts use IRQ 12. A PS/2 touchpad can also be used in lieu of a standard PS/2 mouse. Note that the XTS-300 will also handle a serial mouse which communicates exclusively through a serial port.

3.6.3.5 Parallel Port

The Parallel Port on the XTS-300 is implemented in the ISA compatible National Semiconductor SuperI/O chip included on the motherboard. The LX uses the PC97307 while the GX uses the PC87309. This device features a full IEEE 1284 compliant bidirectional parallel port which supports the Extended Capabilities Port (ECP) including level 2. The SIO chip is implemented as numerous logical devices - the parallel port is logical device #4 on the LX and #1 on the GX. Each logical device is distinct and is addressed separately even though they physically reside on the same chip.

Although supported by hardware, the SPP and EPP modes are not used on the XTS-300. The Extended Capabilities Port (ECP) configuration offers high bi-directional throughput, efficient hardware based handling, including a bi-directional 16-level FIFO with threshold interrupts for both Programmed I/O (PIO) and DMA (ISA) data transfer. A FIFO threshold interrupt can be configured as well as FIFO empty and full status bits, automatic generation of strobes (by the hardware) to fill or empty the FIFO, transfer of commands and data. The XTS-300 uses the ECP FIFO but does not allow DMA operations to take place. It fills the FIFO and will be notified by an interrupt when eight or more bytes are free. While the Parallel Port supports Plug and Play operation allowing its interrupts to be routed on any IRQ from 1 to 15 except for 2 and 13, this feature is not used by the XTS-300 so that the standard IRQ of 7 is always used.

3.7 Peripheral Controllers and Devices

In addition to the main motherboard components, the XTS-300 hardware base also contains other controllers and devices. These are described below:

3.7.1 Multi-Channel Serial I/O Interface (SIO4)

The Multi-Channel Serial I/O Interface card (SIO4) supplies four asynchronous serial ports to the XTS-300. Each serial port on the card has its own UART and support chips. On newer versions of the card, the UARTs are integrated into a single chip. Each UART defines a separate I/O address space used to communicate with the XTS-300.

3.7.2 Video Controller

The XTS-300 video controller is packaged in a single chip which is integrated onto the motherboard. This chip interfaces directly to the PCI bus.

The LX motherboard uses 1 Mbyte of display (video) memory, while the GX motherboard will support 2 Mbytes. This amount of video memory allows standard and VESA high resolutions to be displayed: up to 1280x1024 with 16 colors. All memory access is 32 bit in nature. A FIFO is used to minimize memory contention.

The video controller on the LX motherboard uses the Cirrus Logic chip CL-GD5446 while the GX motherboard uses the CL-GD5480 - both on the motherboard. Both chips do not use downloadable microcode. The video BIOS, which is stored in flash memory on the motherboard along with the system BIOS, is not accessible to XTS-300 programs because the XTS-300 does not allow programs to map that area into their address space. There is no mechanism using the video chip I/O ports or memory to accomplish this. Furthermore, the XTS-300 STOP OS does not make use of the video BIOS itself, although video card initialization and POST tests are run from the video BIOS when the system is powered up or reset.

The video controller is basically an output only device on the XTS-300. Although the controller accepts commands, data flows only in one direction — from the XTS-300 console driver to the video display. Though the console device can be shared by multiple processes, all such processes are at the same mandatory access control level as the console.

The video controller is broken up into four main functional areas: the VGA Sequencer, the Graphics Controller, the CRT Controller (Color/Mono) and the Attribute Controller. Additional components for both the CL-GD5446 and CL-GD5480 integrated chips include: The Hardware Video Window for the simultaneous display of graphics and text, the Video Capture which writes realtime or recorded video from a decoder to the frame buffer for display in a video window, and the Pallete DAC which contains the color palettte and three 8-bit to analog converters. Note that while supported by the chip, a Video Capture hardware interface including V-Port and I2G are not used on the XTS-300 integrated video controller. In addition, although the CL-GD5480 is capable of acting as a bus-master via the video stream engine, this capability is disabled by the XTS-300 by clearing the bus master enable bit in the card's PCI command register which is not accessible outside the Kernel.

The Kernel, or a process which has been granted direct access, can manipulate data on the screen by merely moving/storing directly into the memory-mapped video memory.

The XTS-300 normally uses the video controller in text mode. Specifically, it uses the 80x24 CGA text mode. However, an application can, under controlled circumstances, set the console in graphics mode and thus utilize the Graphics controller.

Normally, both the memory-mapped addresses and the control registers that are mapped into the XTS-300 I/O address space are inaccessible outside of the kernel. This is enforced via the IOPL field (I/O addresses) and the appropriate segment descriptor (memory-mapped addresses) in the kernel. All I/O to the video controller must be done through the normal system gates (e.g., device_io, device_control).

A single process at a time may request exclusive access to both a small subset of the video controller's I/O address space and/or the entire video memory. In that case, the system is careful to ensure that the manipulation done to either is visible only to that process. Such a process can perform I/O directly using either I/O instructions or referencing the video memory. However, whenever communication is requested to

the TCB by the SAK key, the system will always place the console in text mode, back in a known state, and will disable access by untrusted processes to the video hardware.

Note that the video controller does not interrupt the system to signal that it has completed its task. It is theoretically possible to program the controller via jumpers on the board to interrupt at the end of the current active display frame, but this is not done on the XTS-300.

The monitor is an output-only device when used by the XTS-300. The XTS-300 treats it as part of the console logical device, which also includes the keyboard and mouse. The mandatory level of the monitor is the same as the current session level of the user logged into the console.

3.7.3 Mouse

No particular model of mouse is specified for the XTS-300. The mouse need only be either the basic RS-232 serial type or a standard PS/2 mouse connected to the PS/2 port. The mouse must not have a memory of previous mouse activity nor the ability to be programmed to edit the signal send to the host.

The mouse is an input-only device when used by the XTS-300. The XTS-300 treats it as part of the console logical device, which also includes the keyboard and monitor.

3.7.4 SCSI Host Adapter

The XTS-300 SCSI host adapter is packaged in a single chip which is integrated onto the motherboard. This chip interfaces directly to the PCI bus. The host adapter provides an interface between the application and system software and the SCSI "target" devices: hard disks, tape drives, PC card readers, CD-ROM drives, and Data Transfer Cartridge Readers. The host adapter controls data transfers to and from the target devices. Multiple instances of these devices can populate the primary and/or secondary SCSI bus.

The LX motherboard uses the AIC7880 model. The AIC7880 is a highly integrated chip that can perform all the functionality that used to be allocated to multiple subcomponents. The host adapters can manage the SCSI bus asynchronously to CPU operations and thereby serve as a coprocessor to the CPU.

The GX motherboard uses the AIC7896 model. In addition to the characteristics of the AIC7880, the AIC7896 is actually two independent SCSI host adapters on the same chip. These adapters each have their own memory, CPU, buses, and I/O base address space. They share an IRQ, as do all SCSI host adapters on the XTS-300. Each half of this chip is equivalent to the Adaptec 7890 host adapter chip.

Both the AIC7880 and AIC7896 are SCSI-2 compliant allowing "fast", up to 40 Mbytes synchronous transfers over a SCSI bus. The AIC7896 is also SCSI-3 compliant allowing "ultra fast", up to 80MB synchronous transfers over a SCSI bus. Both models and the SCSI bus conform to ANSI X3.131-1986 ("SCSI-1"). They also conform to the Common Command Set standard for SCSI communication: ANSI X3T9.2/85-52. SCSI-2 (ANSI X3T9.2/86-109) is a superset of the other two standards.

Both host adapters supports the wide SCSI bus or a narrow SCSI bus, and will be configured appropriately in the XTS-300 depending on the nature of the device being connected. Narrow devices will be configured to use a narrow bus, and wide devices (mostly hard disks) will be configured to use a wide SCSI bus. Both host adapters support "ultra" speed SCSI bus transfers and "differential" SCSI buses, in addition to "single-ended" SCSI buses. The AIC7896 also supports "ultra2" speed SCSI bus transfers. Note that for

optimal performance, the AIC7896 is normally configured (and has the connectors arranged to support) one "differential" or LVD bus and one "non-differential" or SE bus. LVD devices are to be connected to the LVD bus, and take full advantage of the LVD architecture, while SE devices are connected to the other bus and run at their maximum speeds.

Additional SCSI buses can be configured on both motherboards. To do so, extra cards are installed on the PCI bus. The AIC2940U is used to expand the SCSI subsystem on the LX motherboard using the AIC7880 chipset. Though the 7880 chipset allows use of a wide and/or differential SCSI bus, the 2940U does not support those options. The 2940U2W is used in the GX with the AIC7896 chipset. The AIC2940U2W provides the same functionality as the motherboard versions. However, an external SCSI device cabinet is used to connect the additional bus to the controller. They are connected together via an external SCSI cable. The cabinet provides power and SCSI connectors for up to 15 target devices and accepts a SCSI bus terminator plug. Both the GX and LX models will support a maximum of four SCSI buses total.

The adapter is capable of performing DMA transfer to the system memory. It is also capable of performing scatter/gather functions; however, this function is utilized only within the Kernel. The Kernel does not make scatter/gather functionality available to outer ring software.

There are a number of I/O ports on the adapter and some structures are memory-mapped. The Kernel ensures that these ports and memory-mapped areas cannot be accessed from outside PL0. There are internal registers and RAM on the host adapter, but these are not used by the Kernel. An I/O request will indirectly write some of these locations, but, the locations can never be read by outer ring software.

Microcode exists on the adapter and must be downloaded by the TCB during system startup. The TCB prevents further downloads after that.

Separation of data is dependent upon the Ring 0 software.

3.7.5 Hard Disk

The hard disks used on the XTS-300 are all SCSI devices with asynchronous or synchronous, wide or narrow, ultra or ultra2 interfaces. "Differential" or LVD types are supported as well. The hard disks come in various sizes from 4 Mbytes up to 36 Gbytes. As many disks as can be accommodated on the SCSI bus(es) can be configured. Like other supported SCSI devices, a given device communicates with the system only across the SCSI bus and only communicates with the SCSI host adapter, not with other SCSI devices. The CPU cannot directly access registers and buffers on the drive.

3.7.6 DAT Tape Drive

The DAT tape drive supports 4mm tapes, records in the DDS, DDS-2 or DDS-3 tape format, and can write up to 12 Gbyte of uncompressed data. In addition, it features on-board hardware data compression, allowing it to write and read compressed data. It is SCSI-2 compliant and can perform synchronous data transfers.

The tape drive is designed to allow a firmware download from specially formatted tapes. However, a special firmware build for those units designated for the XTS-300 has disabled this feature. This firmware build was obtained from the vendor and is applied by Wang during assembly of each XTS-300. Further attempts to load firmware result in an immediate eject of the firmware tape.

A SCSI tape drive exists on the XTS-300 as a single level device only. However, privileged backup/restore software on the XTS-300 is capable of writing multi-level data on the tape. It is up to that privileged software to properly identify and label each unit of data so that it might later be properly retrieved and placed at the correct level with the proper security protections.

To help ensure data integrity on the tape drive, parity checking is enabled on the SCSI bus, informing the host of any data corruption problem during a bus transfer. In addition, the drives use three-layer hardware ECC, read-after-write verification and data frame checksums.

3.7.7 Ethernet Controller

The ethernet adapter (Znyx NetBlaster Twisted Pair Lan Adapter) is a PCI board designed to connect the XTS-300 directly to either two (ZX348/ZX348Q) or four (ZX346/ZX346Q) LANs. To accomplish this, the Ethernet adapter uses the IEEE 802.3 10BaseT or 100BaseTx connections.

The ethernet adapter card appears to the system as a PCI bus with four slots. This is accomplished using the 21152 PCI to PCI bridge chip. Each slot provides one ethernet connection using a separate 21140 chip on the ZX346/ZX348 or 21143 chip on the ZX346Q/ZX348Q which occupies one PCI slot each on the ethernet board. Each ethernet connection is also connected to the physical LAN via separate ICS1890Y transceiver chips and XFATM2 transformers on the ZX346/ZX348 or the QS6611 transceiver with the transformers integrated on the RJ45 on the ZX346Q/zx348Q. Each device also has independent clock sources provided by the 21152 controller chip. Each port on the board acts as if it is an independent board and will be referred to as an Ethernet device. The two port board is identical to the four port board except for unpopulated chip positions. Each port has its own memory base and I/O base. The XTS-300 only uses the IRQ and I/O base, there is no reference to the memory base in the software.

The Ethernet controllers do not have on-board BIOSs or firmware. There is no interaction between them and any other peripheral device on the system. Also, the Ethernet controller is a single level device. The device drivers in the operating system handle the separation of data.

3.7.8 PC Card Reader

The PC card reader supports type I, II, or III cards which conform to PCMCIA version 2.1 (though type III cards can only be used in the lower slot and only when no card is in the upper slot). The PC card reader used on the XTS-300 is either a dual slot or eight slot reader which can simultaneously access two PC cards, or eight PC cards respectively.

The reader conforms to the SCSI-2 standard and is a simple SCSI device in that it is a narrow device (i.e., 8-bit bus). All models support asynchronous SCSI bus transfers, while the eight slot model supports synchronous transfer. The PC card reader is unique among the XTS-300 SCSI target devices in that the physical unit accommodates multiple readers and these are addressed using different logical units. The reader also defines two different address spaces withn the slot which are accessed using different LUNs: attribute memory (32 Mbytes) and common memory (64 Mbytes). The two slot version actually assigns different LUNs for the different address spaces. Although each slot (each an LUN pair) are treated as separate, independent devices by the TCB, the two LUNs which are associated with a particular slot are not. This brings the total number of logical units supported by a single two-slot reader to four. By contrast, the eight-slot model uses the different memory address spaces within the slot via a bit setting in the specific read/write SCSI

command block. The TCB device drivers adjust the setting of this bit based on the memory address being targeted.

The firmware cannot be modified using SCSI commands and cannot be downloaded from a PC card. The PCMCIA specification defines structures (such as the CIS) and formats in the address spaces of the PC card, but the TCB is not aware of these and does not attempt to verify them or maintain their integrity.

The only kind of PC card supported by the XTS-300 is a Fortezza card. A Fortezza card is a sophisticated cryptographic device with on-card microprocessor, firmware, ROM, and RAM. A Fortezza card supports many commands, such as encrypt, decrypt, sign, and update keys. The card also protects itself from unapproved access by maintaining PINs and different user personalities.

As is the case for other removable media (i.e., tape, CD-ROM, diskette, and DTCR), the Fortezza card itself is not part of the evaluated hardware configuration. The TCB never uses a PC card for its own purposes (such as encrypting the password file) and does not rely on correct operation of the reader. From the point of view of the TCB, all data on a Fortezza card is at a single level, that level must be externally associated with the card, and the operator must set the level of the PC card reader to that level before the card is inserted. Site procedures are required to ensure that only Fortezza PC cards are used and that Fortezza cards are not removed from the reader while the device is in use (indicated on the front of the device by an LED).

The reader is not involved in trusted path nor I&A from the perspective of the TCB. The fact that the Fortezza card maintains PINs and personality information is irrelevant to the TCB. The TCB does not attempt to correlate user information on the card with information in the XTS-300 trusted databases and the TCB does not provide a trusted path for users to enter a PIN for card access.

3.7.9 CD-ROM Drive

The CD-ROM supported by the XTS-300 is a SCSI-2 device that runs at 6-14X and 17.3-40X speed. It is a removable device with an audio/headphone circuit. This device supports Compact Discs (CDs) with the CD-DA (audio), Red-Book, Photo CD (multi-session) CD-Extra, Yellow-Book, Video CD (MPEG), CD-RW(Read), CD-ROM XA, CD-I, CD-I Ready, CD-R(Read), CD-I Bridge and CD-G.

The CD-ROM is pre-programmed with downloadable firmware. However, in order to be downloaded, the host must use the SCSI Write Buffer command to accomplish this. The XTS-300 does not use this command or allow any application to send it to the device.

A CD-ROM, unlike an XTS-300 hard disk, is a read-only, single level device. The TCB ensures that the device cannot be mounted and that it is a single partition device. A CD-ROM disk is never treated as if it contained a STOP file system. The TCB does not itself rely on correct operation of the CD drive.

There is no direct communication between the CPU and the CD-ROM drive. All communication proceeds through the SCSI host adapter.

The CD-ROM comes with special audio auxiliary controls, a headphone jack and an audio output connector to a sound board or audio amplifier. Since the XTS-300 operates only on CD Type I discs, these controls have no meaning on a STOP operating system and should not be connected or used (which must be guaranteed by site procedures).

3.7.10 Data Transfer Cartridge Reader(DTCR)

The Data Transfer Cartridge Reader (DTCR) device is a drive for a removable cartridge. The cartridge, named a "data transfer cartridge" (DTC), is a specialized device for transferring avionics data to an aircraft before a flight and from an aircraft after a flight. The DTC is a solid-state, random-access device with battery power to preserve data while not plugged in to a reader.

The DTCR conforms to the SCSI-2 standard and is a simple SCSI device in that it is a narrow device (i.e., 8-bit bus), supports only asynchronous SCSI bus transfers, and supports only a single physical unit per SCSI ID (though the physical unit can be accessed via two logical unit numbers (LUNs)).

The operational firmware can be modified using SCSI commands, though this functionality is not allowed by the TCB.

The DTCR partitions memory on the DTC into two regions: a file allocation table (FAT) and a file data area. The structures and formats in these areas are set by applications, not by the hardware or firmware. Access is made to the FAT using LUN 0 and access to the file area is made using LUN 1. Two different processes could open the different partitions, but the TCB software ensures that the two partitions are always at the same MAC level — therefore the DTCR is a single-level device. The TCB never uses the DTCR for its own purposes and therefore does not rely on correct functioning of the device.

3.8 Multiprocessor Architecture and Environment

The XTS-300 APIC controlled bus has the capability to support up to eight independent processors operating in a tightly coupled symmetrical Multi-Processor (MP) system. However, when run under the control of STOP 5.2.E there is a limit of two processors in the flat mode scheme. An APIC is an integrated piece of the CPU. The hardware supports two different kinds of APICs. The local APIC is an integrated component of the CPU chip. At the system level, there is a distinct I/O APIC which is disabled by the XTS-300. All the CPUs in the system are interconnected though the APIC bus. Although the software assigns each CPU a unique number (logical CPUID), the hardware identifies the CPU by another number (physical CPUID) which may be different.

In a system initialization sequence, there is a competition among physically connected CPU to be the boot processor. The software then assigns the boot processor a logical CPU number of zero. The other CPU is then assigned in the order that the boot processor wakes them up. Once STOP 5.2.E is fully initialized, the XTS-300 operates as a fully symmetrical MP system (i.e., all processors are equal, and can perform all operations with all devices).[12]

3.8.1 Memory Organization

All processors share the same physical memory. Furthermore, all processors execute the same copy of the operating system. Controlled access to the system resources is accomplished using spin locks as implemented with read/write interlock instructions, see Section 3.8.1.2.

[12] The one anomaly is that only logical processor #0 handles device interrupts.

3.8.1.1 Intel Cache Memory Considerations

Although the Intel hardware ensures the consistency in an MP environment between the different on-board/external instruction and data caches, special consideration is given by the XTS-300 operating system to ensure that the special page descriptor translation lookaside buffer (TLB) and processor segment shadow registers are updated properly in all CPUs.

3.8.1.2 Read/Write Interlock Instructions

Software race conditions in an MP environment can be controlled by spin-locks if lock words can be set atomically by a given processor. Any processor can serialize access to memory through the use of locked bus cycles. Requests for control of the bus are ignored during locked cycles.

The XTS-300 supports certain memory locking instructions which are needed in order to support tightly coupled MP operations. These instructions are divided between the bit test and change instructions, exchange instructions, and various one/two operand arithmentic and logical instructions. Locking is generally accomplished through the use of the lock prefix for each instruction.[13] Locking is guaranteed only to the level of lock operand (8, 16, 32 or 64 bit chunk), but a larger memory area may be locked The integrity of the lock is not dependent on memory field alignment. Rather, the lock (LOCK#) signal is asserted for as many bus cycles as necessary to update the entire operand.

The L1 and L2 caches are write-back caches. Cache coherency between Intel Pentium II/III processors are maintained by the SNOOPER mechanism on the BIU which detects through snooping that another processor has attempted to access memory location that it has modified but not yet written back to system memory. That SNOOPING processor will signal the other processor through the HITM# signal and send it the contents of the memory location at the same time. It is the responsibility of the memory controller to snoop this signal and update memory.

The bit test and change instructions consist of the following: Bit Test and Set (BTS) and the Bit Test and Reset (BTR) The exchange instructions include the Exchange Register/Memory with Register (XCHG), Exchange and Add (XADD), and Compare and Exchange (CMPEXG). The arithemetic and logical instructions include Increment by 1 (INC), Decrement by 1 (DEC), One's Complement Negation (NOT), Two's Complement Negation (NEG), Add (ADD), Add with Carry (ADC), Integer Subtraction (SUB), Integer Subtraction with Borrow (SBB), Logical And (AND), Logical Or (OR), and Logical Exclusive Or (XOR)

3.8.2 The Interprocessor Interrupt (IPI)

The IPI interrupt mechanism allows one processor to interrupt another processor in order to perform a specific function such as flushing part or all of the target processor's TLB. The IPI interrupt is also used to start the second processor during system initialization. The IPI interrupt acts like an INT instruction on a specific IRQ.[14]

[13] The exchange instructions do not need to have the lock prefix set.
[14] The XTS-300 uses a software generated NMIinterrupt.

3.8.3 APIC Programmable Timer

The APIC Programmable timer is used as a timeslice clock. There is one timer per CPU. When interrupted, the Scheduler process associated with the CPU will take control and perform its scheduling functions.

3.8.4 Locking on the APIC Controlled Bus

Bus masters use the lock bus cycle mechanism to ensure that there is reliable communication between them. The Intel Pentium II/III processor protects the integrity of certain critical memory operations by asserting an output signal called LOCK#. Reads and writes of aligned 64 bit operands and 128 bit instruction prefetches are protected by an output signal called PLOCK#. The processor automatically asserts one of these signals during certain critical memory operations while the software can specify that other select memory operations need to have LOCK# asserted.[15]

Some critical memory operations that automatically assert the LOCK# signal include: acknowledging interrupts, setting the Busy bit of a TSS descriptor, updating segment descriptors,[16] updating page directory and page table entries, and executing an XCHG instruction.

3.8.5 IO

Any processor can issue an I/O request to any device. However, the interrupts on completion will be always handled by the boot processor.[17] Hence, it is possible to initiate I/O on one processor, and field the interrupt on another processor.

3.9 Hardware Initialization

The system BIOS provides a means for booting the XTS-300 (though it is used more generally by some other operating systems to communicate with hardware and vice versa). The BIOS resides in a "flash" memory component off the X bus (which is managed by the PIIX4). This flash memory is non-volatile, but can be written using appropriate command sequences (which is useful for BIOS upgrades or recovery). However, in an evaluated configuration, the OS never writes to flash memory, the OS does not allow untrusted code to write to flash memory, and the site must not allow users to load BIOS from diskette.

The Opsys Loader is loaded by the BIOS and, once loaded, disables the BIOS from being invoked by the system again. In addition, the vector table used to invoke the BIOS is destroyed by the Opsys Loader. For more information on testing and initialization, see Section 7.6, page 110 on system integrity.

The BIOS is logically divided into the following parts:

- General, system BIOS

[15] This is accomplished through the lock instruction prefix.
[16] The STOP 5.2.E also ensures that the segment shadow registers are up to date when one processor modifies a segment attribute
[17] During startup, the boot processor specifies via an APIC register setting that all interrupts will be sent to it.

- Power-on self tests (POST)
- Update recovery code
- Video BIOS
- SCSI host BIOS
- Setup and SCSISelect

3.9.1 Power On Self Test (POST)

POST initializes some data structures, tests system components, and boots the operating system. More specifically, POST writes to the CMOS RAM and the low memory information regarding the system configuration. POST tests the system, the ROM and the RAM. POST also sets the interrupt vector table and checks for peripheral equipment.

To run the POST, one of three methods can be employed; turn the XTS-300 on; press the reset button on the Intel Pentium II/III; or press ⟨control⟩-⟨alt⟩-⟨delete⟩ on the keyboard if DOS is currently booted.

3.9.2 BIOS Setup Utility and Diagnostics and Utility Software

The BIOS setup utility retrieves information about the hard disk drive, floppy drives, monitor, date and time and stores the information in CMOS RAM. The diagnostics and utility software checks the system components, runs off-line, and reports any errors to the user. To run the BIOS setup utility, the ⟨delete⟩ key must be pressed when prompted at system startup. To run the diagnostics and utility software, the XTS-300 must be shut down and a special DOS diskette must be booted.

Setup (SCU/SSU) is a superset of the BIOS "setup" utility. SCSISelect is used to configure the SCSI subsystem. SCU/SSU and SCSISelect are used during hardware assembly to establish a number of hardware parameters.

This page intentionally left blank

Chapter 4

Software Overview

4.1 Introduction

This section provides a general description of the software portion of XTS-300, STOP 5.2.E. The software components of the Trusted Computer Base (TCB) are the Security Kernel, TCB System Services (TSS), and the Trusted Software. There is also an untrusted software component of the supplied system called Commodity Application System Services (CASS). The first portion of the Software Overview is an introduction which briefly describes each of the software components. This is followed by a description of the process environment. Since the process is the major active entity (i.e., that which causes information to flow or changes the system state), the discussion of how the software components fit together to form the process environment is a precursor to the more detailed discussions that follow. The kernel is presented first, followed by a discussion of system initialization. TSS, Trusted Software, and CASS are then discussed in detail, each in its own subsection.

4.2 Software Components

The Security Kernel software occupies the innermost and most privileged ring and performs all Mandatory Access Control (MAC). The kernel provides a virtual process environment, which isolates one process from another. The kernel implements a variation of the reference monitor concept. When a process requests access to an object, the kernel performs the access checks, and, given that the checks pass, maps the object into the process' address space. Subsequent accesses are mediated by the hardware. The Security Kernel also provides I/O services and an Interprocess Communication (IPC) message mechanism. The Security Kernel is part of every process' address space and is protected by the ring structure supported by the hardware.

The TSS software executes in Ring 1. TSS provides trusted system services required by both trusted and untrusted processes. TSS has the responsibility for creation and loading of both trusted and untrusted programs in STOP 5.2.E. TSS converts the flat kernel segment structure into a hierarchical file system. TSS software enforces the Discretionary Access Control (DAC) policy to file system objects and segments.

Trusted Software performs security-relevant functions and executes in Ring 2. Software is considered trusted in STOP 5.2.E if it performs functions upon which the system depends to enforce the security policy (e.g., the establishment of user authorization). Some processes require privileges (see Section 6.3.4, page 92) to perform their functions. An example of a process that requires privileges is the Secure Server, which needs access to the User Access Authentication database, kept at system high access level, while establishing a session for a user at another security level. Figure 4.1 depicts the structure of a trusted process running on XTS-300.

CASS also executes in Ring 2. CASS provides a UNIX-like interface for user-written applications. The purpose of CASS is to make the multilevel security execution environment transparent to software running

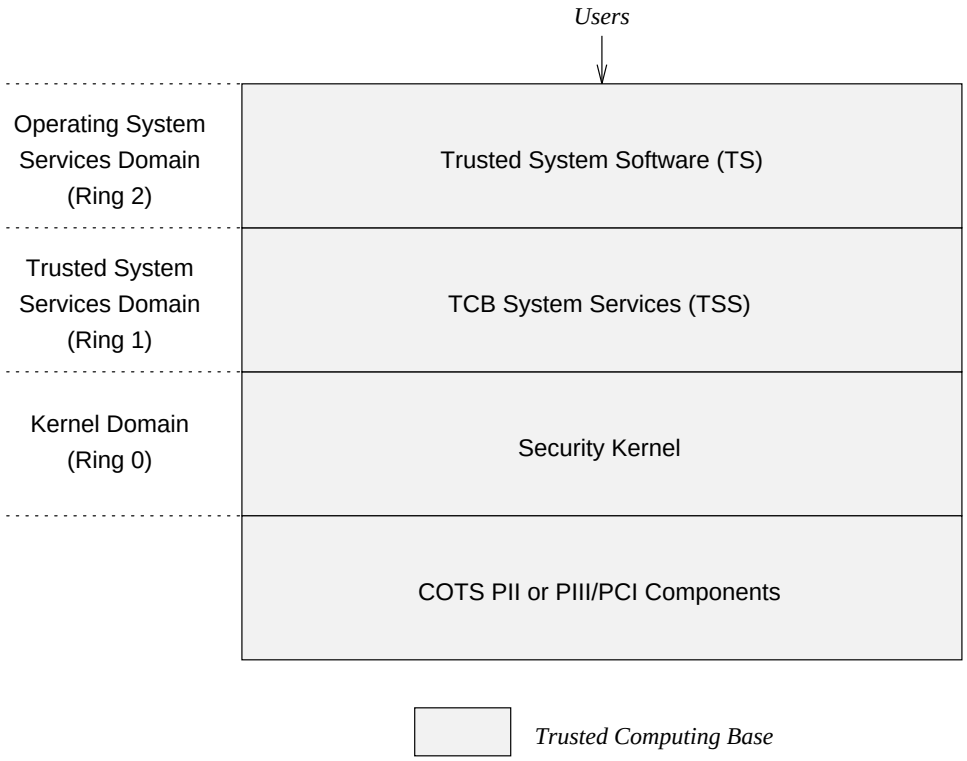

Figure 4.1. XTS-300 System Diagram – TCB Process

in the Application Domain (Ring 3). Figure 4.2 depicts the structure of an untrusted process running on XTS-300.

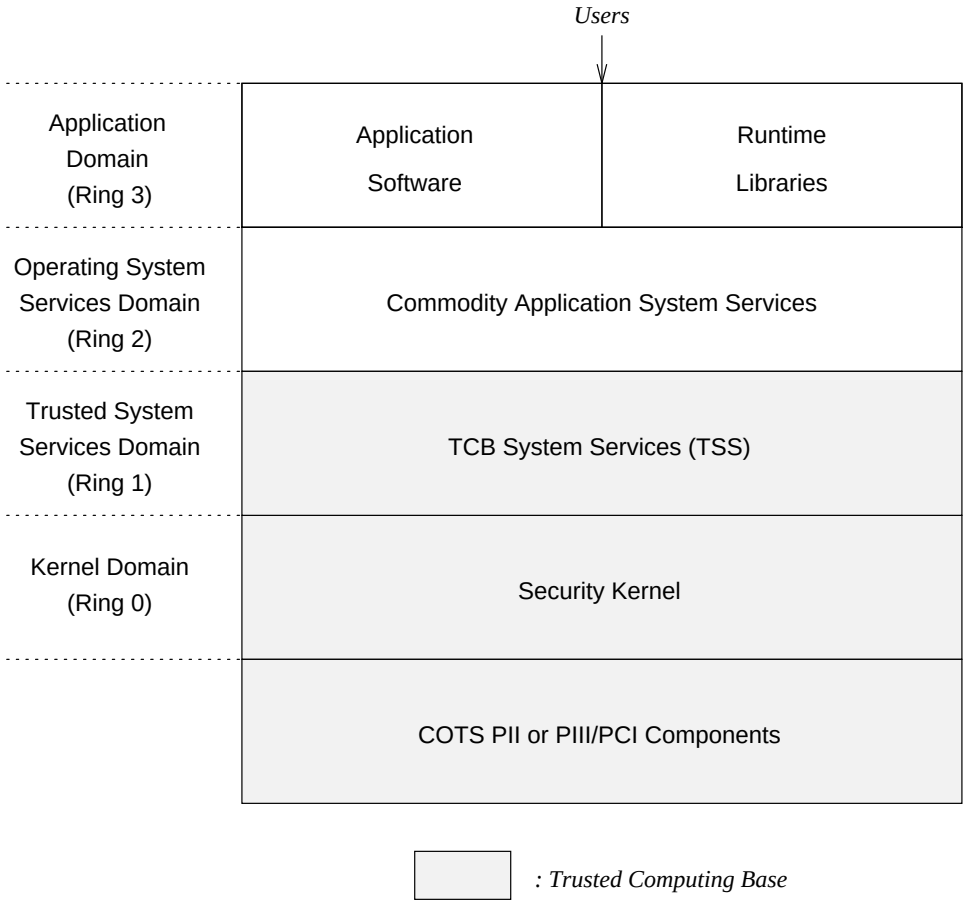

Figure 4.2. XTS-300 System Diagram – Untrusted Process

The virtual address space for a process is depicted in Figure 4.3. The first ten segments are shown to contain global information. These segments are included in every process' address space. All global data segments (kernel declared data, memory map, and the global pool) are accessible only by the kernel. They contain data structures important to the management of the system. The other segments contain the object code for the kernel, TSS, and CASS.

The Binary Compatibility Standard (BCS) segments exist to implement the BCS calling interface for Ring 3 programs and trusted programs running in Ring 2. The memory space for BCS is implemented by special code and data "super descriptors." The two descriptors contain the same base address. The descriptors are separate because the Intel Pentium II/III does not allow both execute and write access to the same segment through the same descriptor. However, the end result is that Ring 2 and Ring 3 processes can access all BCS segments for both execute and write access. The BCS memory space contains the code, static data, dynamic

Proprietary figure removed

Figure 4.3. Process Virtual Memory Address Space

Proprietary figure removed

Figure 4.4. Process Linear Memory Address Space

data, and stack. The BCS segments are completely overlapping. All memory allocated to such segments for a process is potentially available to that process.

However, segment accesses from BCS segments to non-BCS segments is prevented by the hardware and the TCB. The super descriptors refer to distinct regions of the PDIR from those used by the TCB. The kernel prevents allocation of Ring 3 segments below PDIR entry 128 which is beyond all the entries needed by the TCB. Further, a full-size (1024-page) pagetable is allocated for each valid PDIR entry for a BCS segment. Each time an invalid PDIR entry is referenced by a process, a full-size pagetable is allocated during page fault processing.

Transfers to BCS segments from an inner ring are performed by setting the CS register to point to the BCS code segment and by setting the DS and SS registers to point to the BCS data segment.

BCS segments are not mapped from the file. The code and data are copied into memory. As a result, changes to the access of the program file do not affect a process currently executing the file. Access changes to the BCS segments by another process cannot be performed because the BCS segments are treated as temporary segments. Access changes to temporary segments are disallowed by the TCB.

The BCS segments require a contiguous range of at least 512 PDIR entries so that a minimum 2 Gbyte is available for Ring 3 programs. The BCS segments encompass a total of 896 PDIR entries.

As shown in Figure 4.3, the rest of the process address space is private to each process. The kernel, TSS, and CASS all have associated process-local data structures contained in the data segments. Each ring also has an associated stack private to the process. Even if a process is executing in the kernel, it does not have direct access to other processes or objects to which those processes have access. An object must be mapped into the currently executing process' address space in order for the process to access the object.

The Intel Pentium II/III provides for a Paging Directory (PDIR) between the segment descriptors and pagetables to allow greater flexibility in implementing the operating system. The PDIR defines a linear address space. Segment descriptors that define a portion of memory point at PDIR entries instead of pagetables (see Section 3.3.3.1, page 16 for more details on pagetables and segment descriptors). Each process has its own PDIR. Process-local segments point to one, unique PDIR entry and use the same entry that is equal to the segment number. In XTS-300, virtual addresses are translated to linear addresses then translated to physical addresses.

4.3 Process Environment

The virtual process environment on XTS-300 is defined by the four hierarchical domains which are enforced by the hardware ring mechanism. The domain of greatest privilege is Ring 0, that of the kernel. The domain of least privilege is Ring 3, that of untrusted processes. While executing within a domain, a process has direct access only to the resources of that domain and those of less privileged domains. Processes have other attributes which characterize them in addition to the domain in which they execute. MAC labels, integrity levels, privileges, and capabilities are examples. For more information, see Chapter 6.

The environments for each ring are set up by the adjacent ring that is more privileged. When a process is being created, the kernel creates the process-local Ring 0 and Ring 1 environments used for process execution. TSS creates the local Ring 2 environment, and if the process is untrusted, CASS creates the local Ring 3 environment.

The following two types of process environments are supported on XTS-300:

- Untrusted Application Process
- Trusted Application/System Process.

The environment for each type of process is described below.

4.3.1 Untrusted Process Environment

For an untrusted process, the environment consists of four hierarchical domains:

Ring 3 — The Application or User Domain. For user-written applications, this is the only domain available. Runtime library support software for an untrusted process also runs in this domain.

Ring 2 — The Operating System Services (OSS) Domain. CASS software runs in this domain. The primary purpose of CASS is to convert the primitive TCB interface into one usable for applications software. CASS also provides I/O and process control services to user-written applications. CASS is considered to be high integrity system software. As such, the CASS layer provides untrusted operating system services to user-written application software. *It is not part of the TCB.*

Ring 1 — The TSS Domain. The primary purpose of the software in the TSS Domain is to provide the file system hierarchy. TSS also loads trusted and untrusted programs in STOP 5.2.E.

Ring 0 — The Kernel Domain. The Security Kernel portion of the TCB runs in this domain. A subset of kernel services are callable from untrusted software running in Ring 2.

The software in the Kernel, TSS, and OSS Domains is mapped into the address space of all untrusted processes.

The untrusted environment is restricted to the following:

- Execution within the OSS and Application Domains

- Execution with integrity below the operator integrity level
- Execution with no Security Kernel privileges
- Establishment of an untrusted process via the run command or the start_daemon program in Trusted Software or from the system prompt (if programs with an integrity level below operator have been installed via the tp_edit utility).

4.3.2 Trusted Process Environment

For a trusted system or application process, the environment consists of three hierarchical domains:

Ring 2 — The OSS Domain. This domain contains various trusted system functions, collectively referred to as Trusted Software. This portion of the TCB provides specialized security-relevant services.

Ring 1 — The TSS Domain. This domain contains the TSS portion of the TCB.

Ring 0 — The Kernel Domain. The Security Kernel software runs in this domain.

Software in the Kernel and TSS Domains is mapped into the address space of all trusted processes. A degenerate case of a trusted process is a kernel process whose environment consists entirely of the Kernel Domain. For a discussion of kernel processes, see below.

4.3.3 TCB Interface

There are four different mechanisms, or entry points, that provide access to the TCB. The entry mechanisms to the TCB are:

- The Kernel gate that causes a service to be performed on behalf of the caller, followed by a return.
- The TSS gate that also involves performing a service for the caller, followed by a return.
- The Secure Server as invoked by the Secure Attention Key, that establishes a connection with the TCB for the purposes of invoking a trusted process.
- The hardware, which includes instruction execution and traps, exceptions processing, and interrupts from I/O devices.

4.4 System Initialization

System initialization is performed in two phases. The first is performed by the system loader and the kernel startup process; the second is performed by the Secure Startup process in Trusted Software (see Section 4.7.1.1, page 63, for more details).

Proprietary material removed.

4.5 Security Kernel

This section provides a general description of the Security Kernel. The kernel architecture, security-relevant features, and kernel entry mechanisms are discussed as well as other important kernel features. The areas of the kernel discussed in more detail are: segment management, process management, device management, memory management, scheduling, and the support functions.

4.5.1 Security Kernel Architecture

Proprietary figure removed

Figure 4.5. Kernel Hierarchy Diagram

Proprietary material removed.

4.5.2 Kernel Entry and Return

Once initialized, the kernel can only be invoked via the kernel gate, hardware traps, faults, and interrupts. The primary purpose of the kernel gate is to validate access of the caller and to call the appropriate routine to perform the requested function. Since all kernel calls from outer rings are first processed by the kernel gate, kernel calls are referred to as kernel gate functions. The caller supplies the kernel function code and a pointer to an argument list contained in the caller's address space.

Entry into the kernel is restricted through the use of the gate. Access is restricted by defining a gate segment descriptor for the kernel (the same is done for TSS). The gate descriptor defines a procedure entry point, and specifies the privilege level required to enter the procedure. In XTS-300, there is only one true kernel gate. All kernel gate descriptors point to the same location within the kernel (to a routine that sets up registers and builds the argument list for the kernel gate processing routine). The kernel code segment is protected by defining the DPL of the kernel segment to be zero and by allowing only read and execute access while executing in Ring 0. The segment descriptor for the kernel contains this information.

The processing for a gate call is as follows. First, the process is set non-interruptible by the Time Slice Clock (TSC) Manager. The gate function code is then validated by checking to be sure that the supplied code is greater than zero and less than or equal to the maximum number of kernel gate functions in the system (which is currently 55). The address of the routine to be called is obtained through the kernel gate table that contains, for each gate function defined in the kernel, an address, the least privileged ring number allowed to call the gate function, and the number of bytes in the argument list not including the ring of the caller argument. The ring of the caller is verified by hardware prior to entering the kernel. However, since there is only one kernel gate, the DPL for the kernel segment is set to 2. For gates that can only be called from Ring 1, the ring of the caller is verified again by the kernel gate routine by taking the maximum of the CPL and the RPL and comparing that to the ring specified in the kernel gate table. The ring of the caller is obtained by taking the maximum of the CPL and the RPL. The ring of the caller must be less than the ring number specified in the kernel gate table. The arguments for the called routine are copied from the caller's stack into a local buffer. The ring of the caller is appended to the argument list before the call to the gate is performed.

The kernel gate routine copies the argument list from the local buffer into the kernel stack. The kernel gate does not check the form or the content of the argument list. However, the kernel gate knows the number and size of each argument that is expected for each kernel function. The kernel gate calculates the length of the argument list itself. The specified kernel function is then called. Each kernel function validates the arguments passed to it by validating the ring number in each argument pointer. This prevents malicious use of argument list pointers (a pointer to a more privileged ring than the caller).

The only value returned from a kernel function is the error code that indicates an error from the function itself, an error in kernel gate processing, or no error. Prior to returning the error code to the user, a check is made for any outstanding commands or signals and for TSC expiration.

4.5.3 Segment Management

Proprietary figure removed

Figure 4.6. Segment Branch Table Entry (SBTE)

Proprietary figure removed

Figure 4.7. Segment Management Data Structures

Proprietary material removed.

4.5.3.1 The Segment Manager

Proprietary material removed.

4.5.3.2 The ASTE Manager

The Active Segment Table Entry Manager handles all operations on each ASTE. The ASTE Manager maintains an active segment chain, that includes mapped and previously mapped permanent segments and a binary tree that includes permanent and temporary shared segments. The ASTE Manager allocates page tables from the global pool based on the size of the segment being mapped. The main synchronization routine by which the System Sync process[1] traverses the active segment chain is contained in the ASTE Manager. Finally, the ASTE Manager is responsible for releasing any unmapped segments on a filesystem being unmounted. The ASTE Manager's functions are not visible outside the kernel. No kernel gate functions are supported in this module.

The active segment chain processing supports the System Sync Process in the following manner. When the last unmap of a segment occurs, its ASTE is moved from its current position on the chain to the tail. As a result, the least recently used unmapped segments are nearest to the top of the chain and are the first

[1] For more information on the System Sync process, see Section 4.5.3.4.

candidates for removal. Both the ASTE chain and the associated management mechanism are local to the ASTE Manager.

Previously mapped segments exist primarily as a performance enhancement to keep the number of disk I/Os to a minimum. They facilitate segment map and unmap operations for segments that are often used. For a mapped segment, if a branch is modified, a flag is used to indicate the fact for subsequent system synchronization activity. Mapped segments are synchronized upon periodic system sync process activation or by a specific request to do so. If the mandatory attributes of a segment are changed, the segment is written to disk immediately.

Although the number of ASTEs is not actually limited, there is a threshold of the number of ASTEs that will cause a previously mapped segment to be unlinked from the ASTE chain. This threshold occurs when the system is close to exhausting memory or the global pool. The idea behind this threshold is to allow more previously mapped segments when the system is lightly loaded and less when heavily loaded, when memory loading is the driving performance factor. As long as the threshold is exceeded, each create segment call will cause the chain to be searched, looking for a segment to flush. In addition, the last unmap of a permanent segment will cause the same search.

4.5.3.3 The Lock Manager

The Lock Manager supports UNIX file locking functions, including: read-only locks, write locks, locking and unlocking of file extents (portions of segments), blocking and non-blocking lock requests, reading and writing of locks, partial unlocking of locked extents, combining existing locked regions with similar lock requests, deadlock detection, and modification of the current lock type based on the new request type. The Lock Manager allows a blocking lock request to be broken by signals (IPC messages). The Lock Manager cleans up locks during segment unmap, process deletion, and segment (file) deletion operations.

The Lock Manager supports a kernel gate function, callable from Ring 1, to perform locking and unlocking of segment extents. The rest of the functions in this module are not visible outside the kernel.

To perform a lock request, the calling process must be at the level of the segment and have read and write access to the segment. The exception to the mandatory access rule is that if the lock request is for Ring 1 and the request is for a read-only lock, then only read access to the segment is required. The calling process must also have the segment subtype on its subtype list. During an unlock operation, MAC and subtype access checks are not performed since the caller must be the one who obtained the lock.

When a process unmaps a segment, all the lock entries[2] for that process are removed. When a segment is deleted, all the lock entries are returned to the system. Any processes that are blocked on a lock for that segment are awakened.

4.5.3.4 The System Sync Process

The System Sync Process is a system daemon responsible for synchronizing modified segment branch entries and data pages, modified device branch entries (for a discussion of branch entries, see Section 4.5.5.1), and super pages (see Section 4.5.5.2, for details on super pages) for mounted kernel file systems. The System

[2]Lock entries contain the information about a segment, of which a portion may be locked. Lock entries are linked off the ASTE for the associated segment.

Sync Process, which is created during system initialization, runs at an interval specified by a site-configurable parameter, nominally set at two minutes.

During the synchronization process, the SBTE, the continuation blocks, and the associated data pages for each currently mapped segment with a modified SBTE are written to disk. Only those data pages that changed are synchronized. The operation continues until all modified permanent segments have been synchronized. Temporary segments are never synchronized. If a filesystem has not been mounted for read-only access, its super page is synchronized while the System Sync Process is active.

4.5.4 Process Management

Process Management is accomplished by four major modules in the kernel. They are: the Process Manager, the IPC Message Manager, the IPC Message Services, and the Kill Process.

4.5.4.1 The Process Manager

Proprietary figure removed

Figure 4.8. Process Management Data: PDS and PLDS

Proprietary figure removed

Figure 4.9. Active Process Table Entry (APTE)

Proprietary material removed.

4.5.4.2 The IPC Message Manager

The IPC Message Manager provides the interprocess communication mechanism. There are four kernel gate functions supported in this module, all callable from Ring 2 (untrusted software). Two of them are older gates which were retained for backwards compatibility while the two newer gates were enhanced to allow larger and variable size IPC messages as well as the ability for Ring 1 (other than the File System Daemon) to utilize them. There are two gate functions called to send and two gate functions called to receive messages.

An IPC message consists of a message header and text. The message header comprises the ring of the sender and the unique identifier of the process. The text of the message is supplied by the caller to the kernel gate or by an internal kernel routine used to send messages to a process from the kernel.

Messages are sent from one process to another by copying them from the sender to a queue of messages obtained from the global pool. A process's message is linked on the IPC queue of the APTE for the receiving process. Normally, each ring has its own separate message queue. However, for Ring 1 processes, Ring 2 messages such as alarms are placed in the Ring 1 (execution domain of the process) queue. This ensures that

all messages normally sent to the OSS domain are readable by Ring 1 processes. Ring 2 software is notified of queued messages via a bit in the Ring 2 stack which is set by the kernel.

During the message receipt operation, an entry is pulled off the IPC queue of the current process and the message portion is copied into the recipient's memory. During process deletion, all of the entries on the IPC queue for the process are pulled off and released.

4.5.4.3 The IPC Message Services

The IPC Message Services is responsible for providing the low level inter-process communication mechanism. The IPC Message Services actually manages the message queue entry's in the OSS and TSS IPC queues of each process. The queues were discussed above in the Section 4.5.4.2. However, the actual manipulation of the queues is performed by this layer. This separation has been added to clarify locking and usage of portions of the IPC Message module.

4.5.4.4 The Kill Process

The Kill Process is one of the kernel processes. It represents an independent thread of control that performs some of the process deletion functions. During process deletion, the process deletes all of its segments with the exception of the Task State Segment (TSKSS), the LDT segment, the PDIR, PDS, the page table for free pages (from the global pool), and the kernel stack. The Kill Process is then called to finish the job. The Kill Process also returns the APTE to the global pool. The Kill Process is activated during system initialization.

4.5.5 Device Management

Device Management is performed by seven major kernel modules: the Device Manager, the File System Manager, the PCI Manager, the Physical I/O Layer, Device Interface, the I/O Services, and the Interrupt Manager.

4.5.5.1 The Device Manager

Proprietary material removed.

4.5.5.2 The File System Manager

The File System Manager handles the disk blocks allocated to a particular filesystem. The disk blocks are one of three types: data, branch, or continuation. The data blocks contain only segment data; the branch blocks contain an SBTE for each segment defined on the disk or a DBTE for each device defined in the system. Continuation blocks and the contents of an SBTE are discussed in Section 4.5.3, page 48. The File System Manager performs allocation and deallocation of data blocks based on kernel requests; it also performs creation, deletion, and read and write operations on branch information.

None of the File System Manager's functions are callable outside the TCB. The results of several functions are visible to a user or trusted software, such as mounting and unmounting of a filesystem, locking of a filesystem, and deletion of branch information for segments or devices.

A physical disk can be either a single partition or multiple partition device. Each single partition disk is made up of a bootstrap loader, a system bootloader, and the filesystems themselves starting with the superpage. Each multiple partition disk begins with a Master Boot Record (MRB) containing the Partition Table, followed by the partitions on the disk. Each partition is then defined like a single partition disk with a bootstrap loader, a system bootloader followed by the filesystem. Figure 4.10 shows the structure of a physical disk and of a single filesystem.

Proprietary figure removed

Figure 4.10. Disk and File System Structures

If a partition is not bootable (marked active), the areas containing the bootstrap loader and the kernel loader will be present, but unreferenced. For multi-partition disks, the Master Boot Record contains the Master Bootloader and the Partition Table. The Master Bootloader contains executable code that examines the device Partition Table and loads the Bootstrap loader for the partition which is marked active. The Partition Table consists of a linked list of tables.[3]

The first table resides in the first sector of the disk, also called the Master Boot Record (MBR). This table contains entries for the first partitions on the disk, one of which may be an extended partition. Extended partitions resemble a logical disk in that the first sector of each extended partition contains a partition table with four entries, one of which may be another extended partition. Extended partitions are linked together in this way to provide for an unlimited number of partitions per disk.

Partitions may contain STOP file systems or an known collection of data referred to as a USER type.[4] In the latter case, STOP makes no assumptions on the format of the data. The partition can only be accessed as string of bytes (TRUSTED or USER class), and can never be mounted.

STOP file systems can contain a variant known as a SWAP file system. This is similar to the standard STOP file system (FFS) except that it is only used to contain temporary segments that do not have a branch or a directory entry. Since there is no need to keep information across reboots, the information is maintained slightly differently than an FFS file system.[5] Note that a SWAP file system is never manually mounted by an operator. Rather it is automatically initialized and mounted during system initialization and cannot be unmounted.

The filesystems themselves each contain a super page, branch blocks, and data blocks, including continuation blocks. The super page contains specific information about a filesystem. The super page includes the filesystem version number, the minimum and maximum MAC labels for the filesystem,[6] a pointer to where the branch blocks begin, a pointer to where the pool of free data blocks begins, and a mounted indicator

[3] This is identical to the COTS format used by other operating systems on the Intel platform such as DOS, Windows, Unixware and Linux.

[4] Although similar in name, this is not the same thing as a USER class device supported by the XTS-300 kernel partition. The USER partition type is a partition table entry attribute only and is never set by the set_device_class command. Instead it is set at file system initialization via the TRUSTED mkfsys command.

[5] E.g., disk block free list is not maintained.

[6] Swap partitions do not have or need MAC label ranges because of the transitory nature of the objects stored—they will implicitly span the full MAC access range of the boot file system and will not survive a system reboot

(flag). The super page becomes part of the Mount Table Entry (MTE) for the filesystem when it is mounted. The Mount Table contains information about all currently mounted filesystems.

The mounted indicator on disk is set when the filesystem is successfully mounted, and cleared when the filesystem is successfully unmounted. The kernel does not allow a mount of a filesystem in which the mounted indicator is set. This condition occurs if the system crashes while the filesystems are mounted. The operator must perform a check and repair procedure on the disk device before it can be mounted again to restore filesystem integrity (see Section 7.4, page 108, for more details).

4.5.5.3 Device Interface

The Device Interface layer contains only the Device Interface Module. The Device Interface module simply provides a device-independent interface to the specific device drivers from higher layers. It manages no data of its own. The Device Interface layer is positioned above the Physical I/O layer because it calls many of the routines in that layer. The Device Interface layer is used by several of the higher layers.

4.5.5.4 The Physical I/O Layer

The Physical I/O layer of the kernel includes all of the device drivers that perform actual I/O operations on devices. The Physical I/O layer contains drivers for the following devices: console terminal, floppy disk, pseudo-terminal, parallel (printers), ethernet, SCSI CD-ROM drive, SCSI disk, SCSI tape, SCSI PC card, SCSI DTCR, and serial devices (terminals and printers). The Physical I/O layer also contains interrupt processing functions and support routines used by other kernel managers and many of its functions are supported by the hardware.

No functions are callable from outside the kernel in the Physical I/O layer. Direct access to I/O ports and memory-mapped I/O addresses is restricted to the kernel by setting IOPL to zero. As a result, I/O instructions can only be issued from the kernel. The only exception to this is the video memory/registers which can be made available to an application for direct access. The Kernel will allow access to video hardware only if the process already has the console device open. The Kernel can take away access by changing bits in the process' I/O bitmap and by changing the attributes of the segment descriptor used for the video memory pages. The Kernel will only allow direct access by a single process, even though multiple processes may have the console device open.

4.5.5.5 The PCI Manager

The PCI Manager contains all routines for querying and setting the configuration data spaces for PCI buses. This driver maintains a tree of PCI configuration entries for each PCI bus and device on the system. This manager is typically called by the individual device drivers.

The PCI Manager operates on devices that adhere to the PCI standard 2.1. In order to do this, it accesses the PCI Configuration Space registers to obtain configuration information for the appropriate device.

The module maintains a number of lists. The primary list, and the one that will be externally visible is the list of devices found on the PCI bus(es). This list is just a double linked list from the first device found to the last device found. There is also a linked list of PCI buses. These will be linked both to the parent bus

as well as the siblings and children buses. Devices are also linked to siblings and to the bus to which they are attached. These lists will be built during initialization, and then not be changed again. Each device structure will contain the 16 double words of the standard configuration header. It addition, the information about the slot and function numbers of this device are maintained, and a pointer to the bus structure for the bus to which this device is connected.

4.5.5.6 The I/O Services

The I/O Services provides low-level utility routines to the Physical I/O layer. Placing common logic in this module prevents the device drivers from containing redundant code.

The I/O Services layer is positioned below the Physical I/O layer because it services many calls from that layer.

4.5.5.7 The Interrupt Manager

The Interrupt Manager services all interrupts that occur during system operation. The Interrupt Manager is also part of the Physical I/O layer. The sources of interrupts include: peripheral devices, real-time clocks and interval timers upon expiration, APIC timers, keyboards, and non-maskable, or unexpected, interrupts. The Interrupt Manager is entered only via the hardware.

The main data structure associated with interrupts is the Interrupt Descriptor Table (IDT). The IDT contains interrupt gate descriptors for each possible interrupt that could occur on the system. The descriptors provide pointers to the GDT that in turn point to the appropriate TSKSS for the process for which the interrupt is to be processed. The offset in the interrupt gate descriptor points to the beginning of the appropriate interrupt handling routine. Each descriptor also contains a DPL field. The DPL is set to zero in each descriptor. That prevents transfer to the interrupt handler via the descriptor from outside the kernel. The IDT is indexed by an Interrupt Request (IRQ) number that is determined by the physical location of the device.

4.5.6 Semaphores

The Semaphore layer is the lowest of the four layers which implement Kernel objects. It contains only the Semaphore Manager. The Semaphore Manager implements semaphores as TCB objects. It manages lists of semaphores, pending semaphore operations, and semaphore "undo" operations. It is positioned in the layering scheme as it is because it is used by the Process Manager and contains high-level tasks which may require any of the other Kernel services, including auditing, and scheduling.

This layer does not actually need to be below the Device and Segment Managers since it does not actually interact with many of the other Kernel managers.

4.5.7 Memory Management

Memory Management includes handling of physical memory, implementation of the demand paging algorithm, and allocation of memory to the global pool. The first two functions consist of the Memory Manager

and the Pageout Daemon. Responsibility for the handling of the global pool is delegated to the Global Pool Manager.

4.5.7.1 The Memory Manager

Proprietary material removed.

4.5.7.2 The Global Pool Manager

The Global Pool Manager allocates and deallocates blocks of memory from the global pool contained in the sixth segment of main memory. The global pool is used by the kernel to allocate pages to perform I/O and to allocate system data structures, such as a page tables, APTEs and ASTEs. Global pool management is transparent to other rings. There are no kernel gate functions supported by the Global Pool Manager.

During system operation, the Global Pool Manager requests pages of memory from the Memory Manager. Using the bitmap contained in the first page of the global pool segment, the appropriate storage of the desired size will be found and allocated for the caller's use. In addition, multiple contiguous pages can be found easily. The global pool keeps a table of buffer types that are visible externally. These types are mapped into an internal buffer type with an associated size. The internal types are based upon powers of two in size from 16 to 4096 bytes in length. Each page is subdivided into sections equal to the size of the table for which the page was allocated. If no free entry for a table type exists, another page is allocated and subdivided as described previously. If the request for a new page for the Global Pool Manager cannot be filled, an error is returned so the caller can shutdown the system.

The various internal buffer sizes have a limit on the number of pages that can be allocated and kept for each buffer size. During system initialization the Global Pool Manager will compute a threshold for each internal buffer size. When a page is allocated beyond this threshold, the free routine will work to free the extra page. The thresholds are determined by setting aside areas of the global pool segment for particular tasks. They allow an equal number of buffers for each type to be allocated. The limits are used to compute the thresholds but are not enforced.

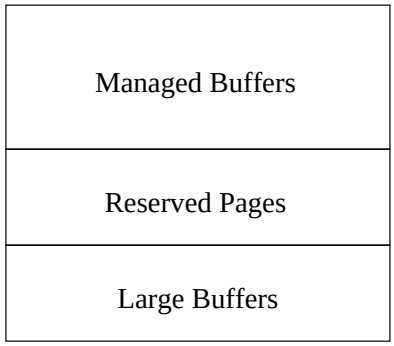

Figure 4.11. Global Pool Usage

Buffers larger than a page in length will always be allocated in a seperate location of the global pool segment and will be rounded up to the nearest page. The large buffers are allocated from the end of the GP segment.

They can use reserve pages. Large buffer pages are freed and returned to the Memory Manager when the buffers are freed. The managed buffers section will be allocated upon demand and the pages never freed. When they take pages from the Reserve, the buffers will be coalesced and the page will be made available for other buffer types when all the buffers within that page are freed.

The maximum size of the total global pool (3072 pages) along with the fact that large buffers can be returned to the Memory Manager helps prevent the system from taking away an excessive amount of memory from the Memory Manager for use by the global pool.

If request for an additional page of memory for the global pool causes the maximum size parameter to be exceeded, the Global Pool Manager returns a failure indication. The caller will then shut the system down with a useful error message.

4.5.8 Scheduling

Scheduling in STOP 5.2.E is done by the Scheduler Processes and the Scheduler Functions that support those processes. Each scheduler process executes within its own address space and is responsible for the dispatching of processes. The Scheduler Functions provide the interfaces for signal and command processing and process execution control. The Scheduler Functions, as part of the kernel, execute within the address space of all processes.

4.5.8.1 Scheduler Process

Proprietary material removed.

4.5.8.2 Scheduler Functions

The Scheduler Functions contain those routines which execute within the process address space that perform operations related to scheduling. The Scheduler Functions provide the signal, command, preemption processing interface, and process execution control interface. There are no gates supported by the Scheduler Functions.

4.5.9 Support Modules

There are a number of miscellaneous modules that support the function of the kernel and other kernel gate functions. They are described below.

4.5.9.1 The Trap Manager

The Trap Manager handles all operational traps. The Trap Manager can correct the condition that caused the trap or fault, and resume execution of the process involved, or pass off the trap to the appropriate handler. If a hardware or software failure occurs within the kernel, the Trap Manager will cause the system to shut down. There are no gate functions supported by the Trap Manager.

For the majority of user traps, the associated data is reflected to the process through the TSS trap interface for processing. For page faults, the Trap Manager will cause the Memory Manager to be activated to bring in the missing page. A page fault[7] is invisible to a process.

Traps are pushed onto the kernel stack. The trap context is placed into a kernel stack frame. The trap procedure is treated as a CALL and executes in the kernel. When the trap processing is complete, the process will resume execution at the location and in the ring of execution specified by the values in the kernel stack frame. If the trap occurred outside the kernel, or further processing in TSS is required, the system resumes execution in TSS by copying the address of the Ring 1 trap handler into the stack frame by which the kernel was entered.

In general, whenever a trap or fault occurs while they are being processed, the system will shut down. There are conditions, however, in which recovery from such conditions is possible (as in the case of page faults).

4.5.9.2 The Audit Process

The Audit Process and the Audit Functions together provide the capability to generate a record of security-related events (the entire audit mechanism is explained in Section 6.6, page 95). The Audit Process serves as the kernel interface to the File System Daemon Process running in the TSS Domain. Audit information is passed to the Audit Process from the Audit Functions module. The Audit Process outputs the information into segments on the filesystem from which the system was booted. The names of the audit segments are passed to the File System Daemon Process for audit files to be produced. The Audit Process is activated during system initialization.

4.5.9.3 Audit Functions

The Audit Functions manage the audit queue and service all kernel requests to generate audit information. The Audit Functions support kernel gate functions, callable from trusted software, to perform auditing of events and to change the audit segment. The audit queue is a linked list of audit frames that contain the link to the next frame, the number of bytes in the frame, the audit data, and a new segment flag. After an audit frame has been filled and placed on the tail of the queue, the Audit Process is awakened to copy the data to the current audit segment. When the copy operation is done, the audit frame is returned to the global pool. Each frame is one page in length. There is a limit of nine audit frames on the queue. When the limit is reached, the audit semaphore is set to prevent additional processes from entering the kernel.

The Audit Functions keep track of the length of the audit segment on disk. If it has been determined that the audit segment is full, a new segment flag is set within the first data frame to be copied to the new segment. The Audit Process checks this flag and if it is set, unmaps the current segment and creates a new audit segment. The name of the new segment is sent via an IPC message to the File System Daemon Process.

4.5.9.4 Metering Management

The Meter Manager provides support for collecting and reporting system performance information. The Meter Manager provides one kernel gate used to report the information that has been collected to a process

[7] Except for page protection traps.

running at operator or administrator integrity level. The gate can also be called to enable certain types of data to be collected. Metering is off by default and cannot be enabled in an evaluated configuration.

The types of data collected include: floppy and SCSI disk I/O times, kernel and TSS execution times, scheduling process execution times, and other I/O data from the File System Manager, the Memory Manager, and the Segment Manager (e.g., branch block reads and writes). Metering is used for system tuning purposes and also as an aid to calculating covert channel capacities.

4.5.9.5 The Advanced Programmable Interrupt Controller (APIC) Manager

The APIC Manager contains the low level routines necessary to: initialize the APIC, start and maintain the APIC timer, send and receive inter-processor interrupts (IPIs), and to boot the application processor. Since the APIC timer exists on a per-CPU basis, these routines are used by the Memory Manager and Scheduler Functions in support of multi-processor operation.

4.5.9.6 The Programmable Interval Timer (PIT) Manager

The PIT Manager provides the interface for obtaining the current value of the timer. The information returned is used as a pseudo-random number for uid generation. The PIT Manager contains a routine to return the time in seconds since 1970. The PIT Manager also services timer interrupts.

4.5.9.7 The Real Time Clock (RTC) Manager

The RTC Manager manages the real time clock. The RTC Manager provides all timer-related services for use by other kernel modules including event handling and covert channel capacity reduction.

The RTC is used by processes to cause an event to happen after a specified period of time. Processes can be awakened or can receive an IPC message for an alarm when the timer expires (established through the execution of a kernel gate). Processes can be suspended, to be awakened later by a user request or by the kernel, if it is determined that a delay is necessary to reduce covert channel capacity. A user request to suspend is performed via a kernel gate which is callable from untrusted software running in Ring 2.

The queue of timer entries is a linked list ordered by the time in which the entries will expire. The first entry contains the time remaining for the first entry. Each entry following the first contains an increment from the immediately preceding entry. The increment indicates the remaining expiration time relative to the preceding entry. The main purpose of this mechanism is to avoid updating every timer entry upon each clock tick.

4.5.9.8 The Bitmap Services

The Bitmap Services define an abstract bitmap management entity as well as routines to implement high-level operations on bitmaps. These services are generic so they can be used by any Kernel module. The data manipulated by these services must be declared by the caller of these services. These services are self-contained so they are placed in the Service Layer. Currently, these services are used primarily by the File System Manager and the Global Pool Manager.

4.5.9.9 The Connection List Manager

The Connection List Manager provides functions for managing segment and device sharing. The Connection List Manager controls the addition and deletion of CLEs. There are no kernel gates supported by the Connection List Manager.

The connection list is examined by kernel modules to locate process information for a particular segment or device. Terminal type (e.g., console, serial, and pseudo-terminal) device drivers use the connection list to manage lists of processes waiting for a particular event to occur on the device.

4.5.9.10 Service Functions

The Service Functions module contains the utility functions common to the other kernel modules; there are no gate functions supported by the Service Functions. The Service Functions include the routines that perform the following operations: check (both MAC and DAC) to determine the caller's ability to access an object in the specified mode, manipulation of semaphores and locks, enabling and disabling of interrupts, and the copying of a data block between an outer and inner ring which includes the validation of a caller-supplied outer ring pointer.

4.5.9.11 Shutdown

Shutdown is the operation by which the system is halted and left in a state such that the system startup process can safely resume operations. The operator can initiate the Shutdown process via Trusted Software, or system shutdown may be invoked by a hardware or software failure encountered in TCB processing. The shutdown kernel gate functions are callable only from trusted software.

During shutdown processing, all the attached terminals are disabled. All non-kernel processes, with the exception of Shutdown itself, are suspended. The operator is prompted as to whether or not a dump tape is to be made. Delete signals are sent to user processes. User processes that are blocked are awakened to perform local termination functions. After the user processes have terminated, the devices and segments allocated to Shutdown are returned to the system. Audit collection is terminated and any existing audit frames are written to the audit segment. The File System Daemon Process is deleted after it has completed any audit segment create operations. Modified data pages, segment branch, and device branch entries are synchronized to disk. Finally, each mounted filesystem is unmounted.

While an orderly shutdown is in progress, all locks held by other processes are honored. That is, the integrity of the filesystem will be held, if possible. However, other internal conditions could result in an immediate system shutdown. Conditions that may cause a "hard" shutdown include an unrecoverable I/O error, an attempt to halt a process that holds a spin lock (the process is in the attempt to update a locked resource while shutdown occurs), a call to shut down the system from a kernel process, a second attempt to halt the system from the process that originally issued the shutdown call, and the original calling process executing at signal level, that indicates the process is trying to delete itself and is unable to do so, or that it will never get to the delete signal to clear its segments from the system.

If an error occurs during shutdown processing, the system is halted and no dump is taken. The mounted filesystems are then left in an unknown state. A check and repair utility must be run on the disks before they can be mounted again. For more details concerning system recovery, see Section 7.4, page 108.

4.5.9.12 The Trace Functions

The Trace Functions provide a mechanism to track Kernel activity. These functions are debug in nature and can be retrieved in two situations: the information is displayed on the system console when the system cannot shut down properly or the system operator dumps the system manually (which is a form of shutdown).

The Trace Functions use a circular array of trace entries for a trace buffer. The trace buffer and associated state variables are only referenced or manipulated within the Trace Functions.

4.6 TCB System Services (TSS)

The TCB System Services occupy the TSS Domain (Ring 1). In contrast to the kernel, most TSS data structures[8] are maintained in process-local memory. The primary functions of TSS are to create and load both trusted and untrusted programs, to provide I/O services to Ring 2 software, and to convert the flat kernel file system consisting of segments into a hierarchical file system consisting of files and directories. The only access control enforcement provided by TSS is discretionary access control to file system objects and segments. TSS is organized into several layers, with functions in the upper layers relying on those in the lower layers. A diagram of the layered organization of TSS is shown in Figure 4.12 (page 82). With the exception of the File System (FS) Daemon and the TCP/IP Daemon, which are both independent processes, TSS executes as part of both trusted and untrusted processes.

4.6.1 Process Management

Proprietary material removed.

4.6.1.1 Program Loader

Proprietary material removed.

4.6.1.2 Kill Manager

The kill handler is notified of all soft kill requests by the kernel. It initiates all necessary TSS cleanup for the process, and notifies the OSS domain of the kill request if that domain has defined a kill handler. To transfer control to the OSS domain, it builds a call frame to place the OSS domain's kill handler address into the return location so that when the TSS kill handler exits, the OSS kill handler will be invoked.

4.6.2 File System

The file system is supported by three layers: FS Daemon, File System, and FS Services. The file system provides the fundamental services needed to support a UNIX-based file system for the OSS domain. TSS

[8] The data structures used by TCP/IP in shared memory are exceptions.

filesystems are supported only on mounted or trusted-class, not user-class, devices. TSS provides a hierarchical file system structure consisting of directories, files, named pipes, and device special files. Although it provides no mandatory access control enforcement,[9] it does provide discretionary access control for file system entities using the conventional owner/group/other permissions, augmented by an access control list. This section first describes the file system structure, and then describes each of the layers.

4.6.3 File System Structure

Proprietary material removed.

4.6.4 File System Layers

Proprietary material removed.

4.6.5 Segment Manager

Proprietary material removed.

4.6.6 Network

Proprietary material removed.

4.6.7 Network Layers

The network is implemented as three layers in TSS. They are Socket Manager, TCP/IP daemon and Socket Services.

4.6.7.1 Socket Manager

Proprietary material removed.

4.6.7.2 TCP/IP Daemon

Proprietary material removed.

[9] MAC policy enforcement is provided by the kernel.

4.6.7.3 Socket Services

The Socket Services layer provides various low-level functions which are needed by both the Socket Manager and TCP/IP Daemon code. This code is largely derived from BSD [36] Unix but like the previous network layers, it has been analyzed and documented so that it could be included in the TCB.

4.6.8 Network Component Communication

Proprietary material removed.

4.6.9 Input/Output

All device input and output is managed by TSS, except for that required to support the kernel's segments. TSS provides callable functions to open and close user devices, and to perform both data transfer and control operations on user devices. TSS also verifies that the device number to which I/O is to be performed is valid.

The support for I/O occurs in several layers in TSS: Entry (Interrupt Manager), I/O Manager, Device Driver, and I/O Services. The I/O Manager handles all TSS gates relating to the management of user devices. It processes user device interrupts and manages I/O buffers. It converts its requests into calls to the appropriate specific device driver; each of these drivers can support the operations on a single TSS device type.[10] The device drivers in turn call the kernel to perform the specified physical I/O operation.[11]

4.7 Trusted Software

Trusted Software executes in the OSS domain; it is invoked by the TSS software or by the user. The user invokes the software by using the SAK, which is implemented as <BREAK> on the keyboard. If the terminal is not currently logged in, the TCB requests the user to login. If the terminal is logged in, the TCB queries the user for a command to execute. Currently executing untrusted software in the OSS and the user domains no longer has access to the terminal; neither does currently executing trusted software in the OSS domain. Trusted software is written to abort upon detection of the terminal loss. If a trusted command was being executed, it is terminated prior to the server prompt.

When the SAK is detected by the serial controller, the line is disabled from further I/O. The enabling of the line requires a privileged operation and is carried out by the Secure Server.

4.7.1 Trusted Processes

Proprietary material removed.

[10] A TSS device type can handle multiple devices. An example of this is the TSS disk type which can handle disks and CD-ROMS.

[11] Sockets are an exception since they do not rely on the kernel for lower level processing.

Proprietary table removed

Table 4.1. Trusted Processes

4.7.1.1 Secure Startup

Proprietary material removed.

4.7.1.2 Secure Initiator

The Secure Initiator is responsible for creating the Secure Server processes as necessary and for managing the pool of free Secure Server processes. It receives a message from the kernel whenever the SAK is struck at a terminal that is not mapped by a process that has the *TERMINAL_LOCK* privilege (e.g., Secure Server). If the key is from the Console, the Secure Initiator sends the message to the Console Server. The Secure Initiator assigns a Secure Server process from the free pool, and forwards the SAK to it. If the Secure Server free pool is empty, a new Secure Server is created. The Secure Initiator also does the same processing for a "hangup" message received from the kernel.

After the processing is completed by a Secure Server, it notifies the Secure Initiator. If the free pool of servers is not full, the Secure Initiator adds the Server to the free pool; otherwise, it asks the Server to kill itself. The Secure Initiator executes at the maximum MAC label and has no privileges.

4.7.1.3 Secure Server

Proprietary material removed.

4.7.1.4 Console Server

Proprietary material removed.

4.7.1.5 Message Daemon

Proprietary material removed.

4.7.1.6 Printer Daemon

The Printer Daemon processes all the print requests for a system printer. There is one Printer Daemon process for each of the system printers.

Each process runs at the maximum MAC label. The Printer Daemon process uses the following privileges:

SET_OWNER_GROUP	to change the owner and group of the Print Daemon to that of the print requestor
SECURITY_STAR_PROPERTY_EXEMPT	to delete lower sensitivity spool files and to map lower sensitivity printers
SIMPLE_INTEGRITY_EXEMPT	to read lower integrity spool files, to map lower integrity printers, and to receive messages from the lower integrity Message Daemon
DISCRETIONARY_ACCESS_EXEMPT	to bypass file subtype checking for trusted databases.

4.7.2 Trusted Databases

Proprietary table removed

Table 4.2. Trusted Databases

Proprietary material removed.

4.7.3 Trusted Commands

This section describes the commands a user can enter after the SAK is pressed. The Secure Server processes the command if it exists in the Secure Server's internal command list. If the command is not in the internal command list, the Secure Server searches the Trusted Program Directory and brings in the trusted program corresponding to the command. For the list of privileges used by the Secure Server, see Section 4.7.1.3. To execute a command, the user must possess the appropriate capabilities; for the list of user capabilities, see Section 6.3.3, page 90.

4.7.3.1 User Trusted Commands

This section describes the commands available to all users. The commands allow a user to manipulate the MAC and DAC attributes for the current session, and to create, attach, and destroy process families at different MAC labels. Table 4.3 provides a list of the commands and required capabilities. The commands are executed either by the Secure Server or by a program at the user's MAC label. The privileges available to the executing programs are also listed in the table. For the list of program privileges, see Section 6.3.4, page 92.

Proprietary table removed

Table 4.3. User Trusted Commands

4.7.3.1.1 Change Command Processor (ccp)

The ccp command allows a user to change his or her command processor. The user must possess the RUN ALLOWED capability to use this command. The Secure Server processes this command.

4.7.3.1.2 Change Default Level (cdl)

The cdl command allows a user to change the default MAC label following login. The user must possess the SET LEVEL capability to use this command. The Secure Server processes this command.

4.7.3.1.3 Change Home Directory (chd)

The chd command is used to specify a new home directory. The Secure Server processes this command.

4.7.3.1.4 Change User Password (cup)

The cup command allows a user or the system administrator to change the user's password. The Secure Server processes this command. The user (including the administrator) must possess CUP ALLOWED capability to use this command.

4.7.3.1.5 Disconnect

The disconnect command is used to disconnect a process family from a terminal. The Secure Server processes this command. The user must possess DISCONNECT ALLOWED capability to use this command. Upon disconnect, the user has no access to the processes in the family. If the processes are still active after the user logs off, they continue to run. Since the user has no access to the process family, commands such as session, kill, ikill, and reattach cannot be used for a disconnected process family. Only the proc_edit command can be used to deal with a disconnected process family.

4.7.3.1.6 Display Free Blocks (df)

The df command reports the number of free disk blocks available on each mounted file system. The command displays information on mounted file systems for which the maximum MAC label is less than or equal to the user's current MAC label. The program has the following privileges:

SIMPLE_INTEGRITY_EXEMPT	to allow high integrity processeses such as operators and administrators the ability to access information on file systems whose maximum MAC label contains a lower integrity.

4.7.3.1.7 File System Manager (fsm)

The fsm command allows a user or an administrator to delete or change the access attributes of a file or a directory, or to create, copy, rename, or display a file or a directory. The Secure Server invokes the fsm

program to process this command. The program executes at the user's MAC label. The program has the following privileges:

DISCRETIONARY_ACCESS_EXEMPT	to bypass the DAC check for the administrator and to bypass file subtype checking for trusted databases
INTEGRITY_STAR_PROPERTY_EXEMPT	to generate audit records and to delete an entry whose integrity is lower than that of the parent directory
SET_DISCRETIONARY_ACCESS	to allow the administrator to change the ACL on an object
SET_LEVEL	to change the file MAC label
SET_OWNER_GROUP	to allow the administrator to change the file owner and group
SET_SUBTYPE_ACCESS	to allow the administrator to change the file subtype
SIMPLE_INTEGRITY_EXEMPT	to allow the operator and administrator to examine and copy lower integrity files
SIMPLE_SECURITY_EXEMPT	to allow examining file system entries at higher sensitivity (e.g., user authentication database)
SECURITY_STAR_PROPERTY_EXEMPT	to delete an entry whose sensitivity label is higher than that of the parent directory.

For a request to change the security attributes of a file, the fsm program ensures that the user is the owner of the file system entry or is running at the administrator integrity level. The user must be at the same sensitivity label and the same integrity label as the file system entry or at the file system entry's sensitivity label and administrator integrity level. For a request to change the MAC label, the kernel ensures that the MAC label is within the range allowed for the file system on which the file system entry resides. The requested MAC label should dominate the parent directory MAC label. If the entry is a directory, the new MAC label must be equal to or less than the MAC label of any files or directories in it.

The fsm program further ensures that only the users with the UPGRADE ALLOWED capability can upgrade an object; similarly, only the users with the DOWNGRADE ALLOWED capability can downgrade an object. In addition, the user without the VIEWING OPTIONAL capability must display the file system entry before downgrading it.

Using the fsm program, operators and administrators have the ability to override the integrity policy of the system. That is to allow operators and administrators to examine and to copy low integrity programs (e.g., to install low integrity programs in the `/trusted` directory).

If the new attributes have either the setuid flag[12] or the setgid flag set, the following conditions will override these flags: if the effective group ID of the user is different from the new group ID in the new ACL, the setgid flag will be reset; if the user or the group ownership of the file is being changed, the setuid and the setgid flags will be reset; and if the new access modes contain any "write" permissions, the setuid and the setgid flags will be reset.

[12] Here setuid and setgid have the same meaning as in UNIX.

4.7.3.1.8 Immediate Kill (ikill)

The ikill command allows a user to immediately terminate all the processes in a process family. The Secure Server processes this command. The user must possess the KILL ALLOWED capability to use this command.

4.7.3.1.9 Kill

The kill command allows a user to initiate the termination of all processes in a specified process family by notifying the processes to terminate themselves. The Secure Server processes this command. The user must possess the KILL ALLOWED capability to use this command.

4.7.3.1.10 Logout

The logout command allows a user to log out of the system. All active processes associated with the user terminal are killed, except for the processes that are disconnected from the terminal. The Secure Server processes this command. The Secure Server releases the terminal and notifies the Secure Initiator to put the Secure Server in the free pool.

4.7.3.1.11 Reattach

The reattach command allows a user to attach the terminal to a detached process family. The Secure Server processes this request and ensures that the user's MAC label and group are the same as those of the process family. If they are not the same, the Secure Server attempts to change the user's MAC level and group to match those of the process family. However, it will do this only after verifying that the proposed level is still within the clearance of the user, the user currently possesses the SL ALLOWED capability, and the proposed group has this user as a valid member. The Secure Server also releases the terminal to the process family and notifies the Secure Initiator to put the Secure Server on the free pool.

4.7.3.1.12 Run

The run command allows a user to initiate execution of a command processor from the terminal. The Secure Server processes this command and makes sure that the user's integrity level is below the OSS integrity level. The Secure Server also ensures that the user has the RUN ALLOWED capability. The Secure Server assigns the terminal a new subtype, releases the terminal to the new process, and notifies the Secure Initiator to put the Secure Server on the free pool.

4.7.3.1.13 Session

The session command allows a user to display the status of the current session on the terminal. The Secure Server processes this request by displaying the user name, group name, terminal major and minor number, MAC label, and the process family information.

4.7.3.1.14 Set Group (sg)

The sg command allows a user to set the group ID for the current terminal session. The Secure Server processes this request and ensures that the user has the SG ALLOWED capability.

4.7.3.1.15 Set Level (sl)

The sl command allows a user to set the MAC label of the current terminal session. The Secure Server processes this request and ensures that the user has the SL ALLOWED capability. The Secure Server also ensures that the new MAC label is dominated by the user's maximum MAC label. The kernel ensures that the new MAC label is within the terminal's minimum and maximum MAC labels.

4.7.3.1.16 System

The system command allows a user to display the system status on the terminal. The Secure Server processes this command. The status includes current date and time, boot device number, boot file system name, system release identifier, and site identifier.

4.7.3.2 Operator Trusted Commands

This section describes the additional commands available to the user with the integrity level of operator or higher. These users can also execute all the commands described in the previous section, except for the run command. The operator trusted commands are executed either by the Secure Server or by a program at the user's MAC label, except for the startup command. The Secure Server invokes the startup program at the maximum MAC label. All of these commands require the user's integrity level to be operator or higher. This requirement is enforced either by the Secure Server or by the Trusted Loader. In addition, the audit and pq_edit commands require the user to be at the maximum sensitivity label (to be able to read maximum sensitivity files: audit files and the prq, respectively). Only two commands require a user capability (SHUTDOWN ALLOWED is required for the dump and the shutdown commands). Table 4.4 and Table 4.5 provide a list of the operator trusted commands and the maximum privileges available to the executing programs.

Proprietary table removed

Table 4.4. Operator Trusted Commands

Proprietary table removed

Table 4.5. Operator Trusted Commands

4.7.3.2.1 Audit

The audit command is used to switch the audit files, to select and to display portions of an audit file, or to remove an audit file. The audit files are at the maximum sensitivity label and the administrator integrity level. The audit program is invoked by the Secure Server to process this command. The audit program restricts the use of this command to the users with maximum sensitivity label and operator or higher integrity level. The program further limits the use of the "remove" and "display" options within the command to the users with the administrator integrity level. The audit program has the following privileges:

SIMPLE_INTEGRITY_EXEMPT	to allow an administrator to read the audit files
DISCRETIONARY_ACCESS_EXEMPT	to bypass checking of file subtype for audit files.

4.7.3.2.2 Check

The check command is used by the operator to check and repair a kernel flat file system. The Secure Server invokes the check program to process the command. The program examines the file system to determine that all storage areas are accounted for and that all allocated areas are mutually exclusive. The allocated areas include the used data blocks, the free blocks list, and the defective blocks list. The program can also clear segment branch entries upon user request. The program sets the "checked" flag (see Section 7.4, page 108).

The program executes at the user's MAC label and requires no privileges. The program ensures that the user is at or above the maximum sensitivity label of the filesystem. The program also ensures that the device on which the filesystem resides is of the TRUSTED class. The kernel ensures that the user is at the same MAC label as the logical device.

4.7.3.2.3 Dump

The dump command allows an operator to dump the memory to a physical device and to initiate system shutdown (see Section 4.7.3.2.12, page 72). The Secure Server ensures that the user's integrity level is operator or higher, and the user has the SHUTDOWN ALLOWED capability. The data is output to the physical device without any MAC checks on the device and without any MAC labels on the output.

4.7.3.2.4 Frestore

The frestore command allows an operator to selectively restore file system objects from a tape to disk. The Secure Server invokes the frestore program to process this request. The program executes at the user's MAC label and has the following privileges:

SIMPLE_INTEGRITY_EXEMPT	to read directories with lower integrity labels
SECURITY_STAR_PROPERTY_EXEMPT	to write directories with lower sensitivity labels
SET_OWNER_GROUP	to change the owner and group of objects being restored

DISCRETIONARY_ACCESS_EXEMPT	to write directories regardless of their discretionary access controls
SET_LEVEL	to modify the object's MAC label
SET_SUBTYPE_ACCESS	to modify the object's subtype.

The program ensures that the class of the source device is USER and the each object to be restored either already exists or can be created on the destination file system.[13] The kernel ensures that user's sensitivity label and integrity label dominate those of the file system objects. This is especially important to the enforcement of role separation. A user at operator integrity cannot restore a file stored at administrator integrity. The kernel also ensures that the user's MAC label is the same as that of the source device.

4.7.3.2.5 Fsave

The fsave command is used by an operator to selectively save file system objects onto a tape from disk. The Secure Server invokes the fsave program to process this command. The program executes at the user's MAC label and has the following privileges:

SIMPLE_INTEGRITY_EXEMPT	to read objects with lower integrity labels
DISCRETIONARY_ACCESS_EXEMPT	to read objects regardless of their discretionary access controls
SECURITY_STAR_PROPERTY_EXEMPT	to update access times of directories that have lower sensitivity labels
SET_OWNER_GROUP	to maintain the labels of lower sensitivity objects when updating the access times of objects.

The program ensures that the source pathname exists and the class of the destination device is USER. The kernel also ensures that the user's sensitivity label and integrity label dominate those of the file system objects. The kernel ensures that the user's MAC label is the same as that of the target device.

4.7.3.2.6 File System Check (fscheck)

The fscheck command allows an operator to check and repair the hierarchical file system. The Secure Server invokes the fscheck program to process this command. The program ensures that all links in the file system are valid, all segments are accounted for and each segment is in exactly one file system object, and that all segments are of proper size. It removes all directory entries to nonexistent file system objects, removes all upward links to nonexistent parent directories, places all unreferenced file system objects in the `/lost+found` directory, removes all unreferenced empty files that cannot be put in the `/lost+found` directory, offers to remove all unreferenced non-empty files that cannot be put in the `/lost+found` directory, removes all data segments with no file control areas, repairs all file size errors, and repairs file control areas that reference nonexistent data segments. It also deletes unreferenced device special files, FIFOs, and empty directories.

[13] If an incompatible object already exists, frestore will not replace it.

The program runs only if the "checked" flag is set (i.e., the check program has been run). The program clears the "mounted" and "checked" flags (see Section 7.4, page 108) to allow the mounting of the filesystem.

The program executes at the user's MAC label and requires no privileges. The MAC requirements are the same as those previously described for the check command. The program ensures that the device class is TRUSTED.

4.7.3.2.7 Make File System (mkfsys)

The mkfsys command is used by an operator to initialize a hierarchical file system.[14] The Secure Server invokes the mkfsys program to process this command. For FFS partitions, the program creates a root directory for the filesystem at the minimum level of the filesystem with the owner and the group IDs of the user. The remaining data blocks are placed on the free blocks list.

The program executes at the user's MAC label and requires no privileges. The program ensures that the logical device containing the file system is of the TRUSTED class. The program ensures that the user is at or above the maximum sensitivity label of the filesystem. The kernel ensures that the user is at the same MAC label as the device on which the filesystem is being created.

4.7.3.2.8 Print Queue Editor (pq_edit)

The pq_edit command allows an operator to edit and display the print queue. The Secure Server invokes the pq_edit program to process this command. The program executes at the user's MAC label and has the following privileges:

SIMPLE_INTEGRITY_EXEMPT	to examine the file or directory of lower integrity to obtain the print file name
DISCRETIONARY_ACCESS_EXEMPT	to examine the requestor's files and directories and to bypass file subtype check for trusted databases.

The kernel ensures that the user is at the maximum sensitivity label.

4.7.3.2.9 Mount

The mount command allows an operator to mount a filesystem. The Secure Server invokes the mount program to process this command. The program executes at the user's MAC label and has the following privileges:

INTEGRITY_STAR_PROPERTY_EXEMPT	to mount a filesystem whose maximum integrity level is higher than that of the user
SECURITY_STAR_PROPERTY_EXEMPT	to mount a filesystem whose minimum sensitivity label is lower than that of the user

[14] Although this is the main purpose of mkfsys, it will also initialize or clear the file system identifier for all partition types as appropriate.

SIMPLE_INTEGRITY_EXEMPT	to mount a filesystem whose maximum integrity label is lower than that of the user.

The kernel imposes the following constraints: The class of the device containing the file system must be TRUSTED. The user's MAC label must be equal or above that of the device and equal to or above the root of the filesystem and the user's integrity must be operator or above. The filesystem minimum and maximum MAC labels must be bounded by the defined system minimum and maximum MAC labels. The mount program ensures that the filesystem has an entry in the File System Name Table.

4.7.3.2.10 Set Device Access (sda)

The sda command allows an operator to change the MAC label and ACL of devices, except for the terminals and line printers. The Secure Server invokes the sda program to process this request. The program executes at the user's MAC label and has the following privileges:

INTEGRITY_STAR_PROPERTY_EXEMPT	to set the attributes of higher integrity devices
SET_LEVEL	to set the MAC label of a device
SET_DISCRETIONARY_ACCESS	to set the ACL of a device
SIMPLE_INTEGRITY_EXEMPT	to obtain the attributes of lower integrity devices
SECURITY_STAR_PROPERTY_EXEMPT	to set the attributes of lower sensitivity devices.

The sda command ensures that the user's sensitivity label dominates that of the device.

4.7.3.2.11 Set Device Class (sdc)

The sdc command allows an operator to change the class of a disk device to one of the following types: mounted, trusted, user. The Secure Server invokes the sdc program to process this command. The program executes at the user's MAC label and requires the following privileges:

SIMPLE_INTEGRITY_EXEMPT	to obtain the device class of a lower integrity device.
SECURITY_STAR_PROPERTY_EXEMPT	to look for a STOP file system identifier on a lower sensitivity devices.
INTEGRITY_STAR_PROPERTY_EXEMPT	to look for a STOP file system identifier on higher integrity devices

The sdc command ensures that the user's sensitivity label dominates that of the device.

4.7.3.2.12 Shutdown

The shutdown command allows an operator to initiate system shutdown. The Secure Server calls the kernel gate shutdown to process this command. Before calling the kernel gate, the Secure Server ensures that the user's integrity level is operator or higher, and the user has the SHUTDOWN ALLOWED capability.

4.7.3.2.13 Set Time (st)

The st command allows an operator to set the system clock. The Secure Server processes this command and ensures that the user's integrity level is operator or higher.

4.7.3.2.14 Start Daemon (start_daemon)

The start_daemon command allows an operator to activate processes that have been configured using the daemon_edit command. The start_daemon command executes at the user's MAC label and has the following privileges:

DISCRETIONARY_ACCESS_EXEMPT	to send a message to an executing daemon process that has discretionary access permissions on messages sent to the process
INTEGRITY_STAR_PROPERTY_EXEMPT	to open the Daemon Attribute database from an integrity level below the maximum and to start a higher integrity daemon process
SET_OWNER_GROUP	to kill a daemon process that has an owner different from the current user of the command and to start a daemon process with a different owner and/or group from that of the current process
SIMPLE_INTEGRITY_EXEMPT	to check for the existence of a daemon, to kill a lower integrity daemon, and to start a lower integrity daemon
SIMPLE_SECURITY_EXEMPT	to check for the existence of a daemon and to kill a higher sensitivity daemon
SECURITY_STAR_PROPERTY_EXEMPT	to open the Daemon Attribute database from a security level above the minimum and to start a lower sensitivity daemon process.

Before invoking the start_daemon command, the Secure Server ensures the user's integrity level is operator or higher.

4.7.3.2.15 Startup

The startup command allows an operator to notify the system to start processing SAKs from other terminals (i.e., other than the console). The Console Server processes this command by invoking the startup program only if the "startup" flag in the Trusted Information Database is set to "false." The flag is set to "false" by the Secure Startup process and is set to "true" by the startup program. This mechanism prevents the TCB from executing multiple startup commands. The program enables the terminal lines, so that the SAK key processing is performed when the user depresses it, putting the system in the multi-user mode. Since the startup program executes with the *TERMINAL_LOCK* privilege, the SAK messages during its execution

are sent to it by the kernel. The program sends these messages to the Secure Initiator for processing. The startup program executes at the maximum MAC label and has the following privileges:

DISCRETIONARY_ACCESS_EXEMPT	to map a device
SET_OWNER_GROUP	to set ownership of the current process
SIMPLE_INTEGRITY_EXEMPT	to map a lower integrity device
SECURITY_STAR_PROPERTY_EXEMPT	to map a lower sensitivity device
TERMINAL_LOCK	to map a locked terminal.

Before invoking the startup program, the Secure Server ensures that the user's integrity level is operator or higher.

4.7.3.2.16 Stop Daemon (stop_daemon)

The stop_daemon allows an operator to terminate a daemon process that has been configured with the daemon_edit command. The stop_daemon command executes at the user's MAC label and has the following privileges:

DISCRETIONARY_ACCESS_EXEMPT	to send a message to an executing daemon process that has discretionary access permissions on messages sent to the process
INTEGRITY_STAR_PROPERTY_EXEMPT	to open the Daemon Attribute database from an integrity level below the maximum and to start a higher integrity daemon process
SET_OWNER_GROUP	to kill a daemon process that has an owner different from the current user of the command
SIMPLE_INTEGRITY_EXEMPT	to check for the existence of a daemon and to kill a lower integrity daemon
SIMPLE_SECURITY_EXEMPT	to check for the existence of a daemon and to kill a higher sensitivity daemon
SECURITY_STAR_PROPERTY_EXEMPT	to open the Daemon Attribute database from a security level above the minimum.

Before invoking the stop_daemon command, the Secure Server ensures the user's integrity level is operator or higher.

4.7.3.2.17 Unmount

The unmount command allows a user to unmount a filesystem. The Secure Server invokes the unmount program to process this command. The program executes at the user's MAC label and has the following privileges:

INTEGRITY_STAR_PROPERTY_EXEMPT	to unmount a filesystem whose maximum integrity level is higher than that of the user
SECURITY_STAR_PROPERTY_EXEMPT	to unmount the filesystem of lower sensitivity
SIMPLE_INTEGRITY_EXEMPT	to unmount the filesystem of lower integrity.

The kernel imposes the following constraints: The class of device containing the filesystem must be MOUNTED, i.e., the filesystem must be mounted. The user's MAC label must equal that of the device and of the root of the filesystem. Alternatively, unmount is permitted if the user's sensitivity label is equal to that of the device and that of the root of the filesystem, and the user's integrity label is operator or above.

4.7.3.3 Administrator Trusted Commands

This section describes the additional commands available to the user with the integrity level of administrator. These users can also execute all the commands available to the untrusted users and to the operators, except for the run command. The administrator trusted commands are executed either by the Secure Server or by a program at the user's MAC label. All of these commands require the user's integrity level to be administrator. This requirement is enforced either by the Secure Server or by the Trusted Loader. None of the commands requires any user capability. Table 4.6 provides a list of the administrator trusted commands, required MAC labels, and the maximum privileges available to the executing programs.

Proprietary table removed

Table 4.6. Administrator Trusted Commands

4.7.3.3.1 Configuration Editor (config_edit)

The config_edit command allows an administrator to edit the following system configuration related databases: configuration, logical device data, printer information, and terminal configuration. The Secure Server invokes the config_edit program to process this command. The program executes at the user's MAC label and requires no privileges. The MAC policy dictates that the user is at the minimum sensitivity and the maximum integrity label. Adding or removing logical devices initiates system shutdown. Changes to the databases take effect upon reboot, except that the changes to a terminal entry in the terminal configuration database take effect when SAK is hit on that terminal.

4.7.3.3.2 Cancel Terminal Lockout (ctl)

The ctl command is used by the system administrator to reenable a locked-out terminal. The Secure Server processes this command and ensures that the user's integrity level is administrator.

4.7.3.3.3 Daemon Attributes Database Editor (daemon_edit)

The daemon_edit command allows an administrator to read and modify the Daemon Attribute database. The daemon_edit command executes at the user's MAC label and requires no privileges. Before activating the command, the Secure Server ensures that the user's integrity level is administrator.

4.7.3.3.4 File System Name Table Editor (fsnt_edit)

The fsnt_edit command allows an administrator to edit the file system mount entries in the root directory of the boot file system. The Secure Server invokes the fsnt_edit program to process this command. The program executes at the user's MAC label and requires no privileges. The MAC policy dictates that the user be at the minimum sensitivity and maximum integrity label.

4.7.3.3.5 Group Access Database Editor (ga_edit)

The ga_edit command allows an administrator to edit the Group Access Authentication and Group Access Information databases. The Secure Server invokes the ga_edit program to process this command. The changes to the database take effect at the next login and do not affect the current users operating in a modified or deleted group. The program executes at the user's MAC label and has the following privileges:

SECURITY_STAR_PROPERTY_EXEMPT	to update the Group Access Information database
DISCRETIONARY_ACCESS_EXEMPT	to bypass file subtype checking for trusted databases.

The kernel ensures that the user is at the maximum MAC label since the group access authentication database is at that label.

4.7.3.3.6 System Parameter Editor (param_edit)

The param_edit command allows an administrator to edit the Trusted Information and Audit Profile databases. The Secure Server invokes the param_edit program to process this request. The program executes at the user's MAC label and has the following privilege:

DISCRETIONARY_ACCESS_EXEMPT	to bypass checking of file subtype for the audit profile database.

Since the database is at the minimum sensitivity and the maximum integrity MAC label, the MAC policy requires that the user be at the same MAC label.

4.7.3.3.7 Process Editor (proc_edit)

The proc_edit command allows an administrator to display and terminate processes in the system. The Secure Server invokes the proc_edit program to process this request. The program executes at the user's MAC label and has the following privileges:

SET_OWNER_GROUP	to terminate the processes of other users
SIMPLE_INTEGRITY_EXEMPT	to obtain the status of processes with lower integrity labels
SECURITY_STAR_PROPERTY_EXEMPT	to terminate a process with lower sensitivity labels.

4.7.3.3.8 Security Map Editor (sm_edit)

The sm_edit command allows an administrator to edit the security map database. The Secure Server invokes the sm_edit program to process this command. The program executes at the user's MAC label and requires no privileges. The kernel ensures that the user is at the minimum sensitivity and the maximum integrity label. Aliases (e.g., short names) for MAC labels can also be established through the sm_edit command. Each alias must be unique.

4.7.3.3.9 TCP/IP Configuration Editor (tcp_edit)

The tcp_edit command permits a system administrator to edit the trusted TCP/IP configuration database and the daemon-specific databases. Note that for TCP/IP to be operational, config_edit must be used to configure the associated ETHERNET device and daemon_edit must be used to configure the associated TCP/IP daemon. The program executes at the user's MAC label and requires no privileges. Before activating the command, the Secure Server ensures that the user's integrity level is administrator. The MAC policy dictates that the user be at the minimum sensitivity and maximum integrity label.

4.7.3.3.10 Trusted Program Directory Editor (tp_edit)

The tp_edit command allows an administrator to edit the Trusted Program Directory databases (`/trusted`) and to modify the security attributes of programs in the `/system` directory. The administrator can also use the command to change the privileges and the integrity label of a program. The Secure Server invokes the tp_edit program to process this command. The program executes at the user's MAC label and has the following privileges:

SET_LEVEL	to set the mandatory access level of a replacement file to the level of the file being replaced
SET_OWNER_GROUP	to set the ownership and discretionary attributes of a replacement file to those of the file being replaced
SIMPLE_INTEGRITY_EXEMPT	to read lower integrity program files

SECURITY_STAR_PROPERTY_EXEMPT	to alter the security attributes of programs in the `/system` directory.

The MAC policy dictates that the user be at the sensitivity level of the program file and at the maximum integrity label. The program does not allow the *MODIFY_PRIVILEGE* privilege to be assigned to any program. The program allows privileges to only those programs with the integrity level of operator or higher.

4.7.3.3.11 User Access Database Editor (ua_edit)

The ua_edit command allows an administrator to modify the User Access Authentication and the User Access Information databases. The Secure Server invokes the ua_edit program to process this command. Removing a user entry does not affect the current session of the user or any objects owned by the user until the SAK is pressed. When the SAK is processed, the Secure Server will log out the user. When a user's clearance is lowered or changed to a level incomparable to the current one, the administrator is directed by a procedure documented in the TFM to delete all of the current user's processes and to delete all of the files to which the user no longer has access. The program executes at the user's MAC label and has the following privileges:

SECURITY_STAR_PROPERTY_EXEMPT	to update the User Access Information database.
DISCRETIONARY_ACCESS_EXEMPT	to bypass file subtype checks for the trusted databases.

The MAC policy dictates that the user of this command be at the maximum MAC label.

4.8 Commodity Application System Services (CASS)

CASS provides an environment on the XTS-300 sufficient to allow the execution of UNIX-based applications; that is, CASS provides a UNIX-like interface to the user. Additionally, CASS provides extensions to allow application software to utilize the multilevel secure execution environment. Thus, CASS provides the necessary services to support both existing UNIX-based software and new applications that are specific to the XTS-300. XTS-300 provides an environment that complies as closely as possible with specifications for UNIX System V, Release 3.0 [3], the American National Standards Institute (ANSI) Standard for the Programming Language C [2], and the IEEE P1003.1 Portable Operating System Interface for Computer Services (POSIX) [7] and the Intel Binary Compatibility Standard (BCS) [8] .

To reduce the size of the TCB, the services provided by CASS are not, in general, provided by the TCB. Whenever possible, the execution environment furnished by CASS is provided by either untrusted services operating in the OSS Domain, utilizing the more primitive underlying TSS and the Security Kernel, or by application runtime libraries operating in the Application Domain. Though CASS and Trusted Software both execute within the OSS Domain, only one of the two will execute during the lifetime of a particular process. There is never a transfer of control between the two. Within the OSS Domain, only CASS executes in an untrusted process and only Trusted Software executes in a trusted process.

The untrusted CASS software is included in each process's memory address space to allow most process creation operations to be simpler and quicker. This means that the hardware does not prevent the OSS

Domain software in a trusted process from transferring control to CASS. CASS software is never needed by a trusted process and, since it is not trusted, must not be used by a trusted process. OSS Domain software in a trusted process is trusted and will never use the segment of a process's memory address space that is reserved for CASS Text (segment 5).

When XTS-300 is executing an untrusted commodity application process, the OSS Domain is outside of the TCB and contains the CASS software which converts the primitive TCB interface to an application-usable interface. CASS is responsible for providing I/O services and process control services to commodity applications. Although untrusted, CASS is considered high-integrity system software. That is, the CASS layer provides untrusted operating system services to application software.

CASS runs in the OSS Domain. Application programs started by CASS run in the Application Domain at an integrity level below OSS. Neither CASS nor user application programs can run with privilege.

4.8.1 Invoking CASS

CASS supports 158 main shell commands and 20 internal shell commands.

CASS is entered via the `enter_cass_gate`. CASS processes the gate request for function code validation via the function `process_cass_gate_request`, obtaining the gate function pointer, and saving the environment for gates that can be interrupted by signals. A User Domain process calls the gate function via the `call_gate_function`. On return from the gate, the IPC queue is drained and a check is made for signals to be processed.

CASS can also be invoked by the Secure Server `run` command which causes CASS to be loaded by TSS `load_process`. CASS is entered via the `cass_entry_point`. This causes the CASS managers to be initialized. The user information is then extracted from the User Access Information database and a terminal control process is created. The default command processor is then started.

4.8.2 CASS Environment Components

The CASS environment consists of runtime libraries that operate in the User Domain, and system calls that execute in the OSS Domain of the XTS-300. These components include the following:

- Runtime libraries to provide UNIX equivalent subroutine calls
- UNIX equivalent system calls
- XTS-300 specific system calls.

4.8.2.1 File System Services

The File System Services pass information to TSS which performs the necessary policy enforcement prior to returning the information to CASS. CASS routines use the hierarchical file system provided by TSS. The MAC and DAC policies of the system are always enforced to ensure that CASS is unable to violate the security policy. CASS is claimed by Wang to be nonmalicious code.

An application requesting the creation of a file system object must be at the MAC label of the new object's parent directory and must possess write permission to the directory as defined by its ACL. When a file system object is created, its MAC label is identical to the MAC label of the creating process; its access modes are equal to the logical AND of the requested modes and the ones complement of the *umask* value for the application. Access modes are mapped into the ACL by TSS.

4.8.2.2 Input/Output Services

The CASS I/O system services function in the same manner as the respective UNIX system calls. With the exception of the *ioctl* system service, which provides an interface that allows processes to control character devices, I/O system services treat all file system objects as regular files.

4.8.2.3 Process Control Services

The Process Control Services provided by CASS perform process initialization and termination operations on behalf of the application. Additionally, CASS Process Control Services are responsible for IPC Message Management, Kill Management, Trap Management, and the management of process-related information obtained from the TSS or Kernel Domains.

4.8.2.4 IPC Message Management

CASS is responsible for the management of all IPC messages directed to the process. It makes the receipt of all messages transparent, except for those messages to be delivered to the application via signals as specified by the *signal* system service.

4.8.2.5 CASS Kill Management

CASS is responsible for handling the deletion of the OSS Domain portion of user processes. Process termination is accomplished by the kernel for the process environment not handled by CASS. Standard UNIX kill notification is used to inform the user of the process deletion.

4.8.2.6 CASS Trap Management

CASS is responsible for handling any trap information passed from the TSS Trap Manager to CASS. These include both external traps and internal traps. External traps are those caused by software executing in the User Domain and internal traps are those caused by CASS software executing in the OSS Domain.

CASS makes the receipt and handling of internal traps transparent to the User Domain except when the trap will result in process termination. External traps are made transparent to the untrusted application, except for those traps to be delivered to the application via signals in the manner specified by the *signal* system service.

4.8.3 Interface Requirements

CASS interfaces with the kernel (Ring 0) and TSS (Ring 1) through gates callable from untrusted software. CASS has no interface to and does not interact in any way with Trusted Software. Untrusted applications running in Ring 3 must interface with CASS to invoke services provided by the TCB.

Proprietary figure removed

Figure 4.12. TCB System Services Layering Diagram

Proprietary figure removed

Figure 4.13. Overview of the STOP File System

Proprietary figure removed

Figure 4.14. XTS-300 Network Components

Chapter 5

TCB Protected Resources

The primary purpose of a Trusted Computing Base (TCB) is to mediate the data flow between, and to provide protection of, selected entities in a computing system. Entities in a computing system fall into two classes: active and passive. Active entities are those pieces in a computing system that do things – they cause information to flow or change the system state (for example, processes or executing device driver code). Passive entities are those pieces of a system that logically contain or receive information (for example, files, devices, memory segments, etc.). Note that an entity may have both active and passive aspects (for example, a device).

The active and passive entities that are under the control of the system security policy as enforced by the TCB are called subjects and objects. Subjects correspond to active entities and objects to passive entities, all under control of the TCB. The target rating for a system determines the extent to which identified subjects and objects must be protected by the TCB; at the B3 rating, all identified subjects and objects must be protected.

The following sections enumerate the subjects and objects of STOP 5.2.E, and discuss their attributes and life cycles.

5.1 Subjects

STOP 5.2.E supports only one type of subject: a process.[1]

5.1.1 Subject Attributes

For each subject, in addition to the current ring of execution, the system maintains the following security-relevant information in the kernel Active Process Table:

- The real user and group identifiers (IDs). This identifies the user and group responsible for the subject.
- The effective user and group IDs. This identifies the user and group on whose behalf the subject is operating. It is the effective user and group IDs that are used in the discretionary access checks.
- The clearance of the user on whose behalf the subject is operating.
- The Mandatory Access Control (MAC) label (i.e., sensitivity and integrity labels) of the subject.
- The object subtypes (see Section 6.3.2, page 89) to which the subject has access.

[1] This definition is independent of the domain of execution of the process, which may change during the process's lifetime (for example, as the process transitions between hardware rings). In STOP 5.2.E, as long as the process is active, it is a subject.

- The effective privileges (see Section 6.3.4, page 92) of the subject.
- The maximum privileges of the subject.

5.1.2 Subject Creation and Destruction

Subjects can only be created by other subjects.[2] At the TCB interface, subjects are created via the `load_process` and `fork_process` TCB System Services (TSS) gates. When `load_process`, is used, the kernel creates a new subject environment that executes within the TCB until the TSS program loader relinquishes control. With `load_process`, the new process environment is determined solely by the attributes of the program file. If the program being loaded has an integrity level greater than or equal to operator, the subject continues to execute within the TCB when loading is complete (i.e., it is a "trusted" subject). If the program is of an integrity level below operator, the subject executes outside the TCB when loading is completed (i.e., it is an "untrusted" subject). `Fork_process`, the other subject creation path provided to untrusted code, creates a new subject whose environment is identical to that of the parent subject. In this case, the new subject executes within the TCB until `fork_process` terminates; execution then resumes at the same point in parent and child. In all cases, the TCB interface provides trusted subjects with the ability to create both trusted and untrusted subjects; however, it restricts untrusted subjects to the creation of untrusted subjects. Details on subject creation and the address space provided to a subject may be found in Section 4.6.1, page 60. Subjects are destroyed by the `release_process` gate.

5.2 Objects

Proprietary material removed.

[2] The first subject in the system (the system loader) is created by the bootstrap loader.

Chapter 6
TCB Protection Mechanisms

6.1 Introduction

The most important services provided by a Trusted Computer Base (TCB) are its protection mechanisms. These mechanisms serve to enforce the system policy, provide separation of address spaces, provide protection of trusted code, and provide accountability for actions taken by subjects; they are implemented using a combination of hardware and software. The hardware mechanisms have already been discussed; the software mechanisms concentrate on the enforcement of policy and accountability.

STOP 5.2.E uses a hybrid security policy model that combines the Bell and LaPadula model [4] for sensitivity and the strict Biba model [5] for integrity into a model wherein the conditions of both models must be met for access to be granted. One benefit of this approach is that it allows STOP 5.2.E to easily support least privilege through the controlled use of integrity levels. This is discussed in more detail on Section 6.2.1.1.

The following sections explore each of the mechanisms provided by software. The discussion will begin with a look at the mechanisms that enforce the mandatory and discretionary policies of the system. This will be followed by a discussion of the secondary mechanisms provided (subtypes, capabilities), as well as a discussion of the privilege mechanism. The focus then turns to accountability, with a discussion on the identification and authentication mechanism of STOP 5.2.E, followed by a discussion on the special set user ID protection mechanism and the audit facilities. The last mechanism discussed will be object reuse.

6.2 Policy Enforcement Mechanisms

The primary protection mechanisms in STOP 5.2.E are those that enforce the mandatory and discretionary access control policies. This section begins with a general discussion of the Mandatory Access Control (MAC) policy of STOP 5.2.E, and how it is enforced. This is followed by a corresponding discussion for the Discretionary Access Control (DAC) policy. After that, the specific mechanisms used for each of the STOP 5.2.E objects are presented, including discussions on: where policy is enforced for that object in the system, how a user can change the attributes of the object, and what happens as a result of changes to the attributes of the object.

Proprietary figure removed

Figure 6.1. Hardware and Kernel Access Checks

Proprietary figure removed

Figure 6.2. TCB System Services (TSS) Access Checks

6.2.1 Mandatory Access Control Policy

The MAC policy grants access based on labels assigned to subjects and objects. In order for access to be granted under the MAC policy, the label on the object must satisfy a specific relationship, determined by the type of access needed, to the label of the subject requesting access.

6.2.1.1 MAC Labels

All subjects have associated with them a MAC label that represents the current sensitivity and integrity labels of the subject. The clearance of the user associated with the subject must always dominate the current MAC label of the subject. This restriction is enforced by the kernel. Furthermore, all objects unambiguously have associated with them a label that reflects the sensitivity and integrity of the information in the object.

MAC labels contain the following information:

- Sensitivity label:
  - Sensitivity level (16 hierarchical)
  - Sensitivity categories (64 nonhierarchical)
- Integrity label:
  - Integrity level (8 hierarchical). STOP 5.2.E has predefined meanings for the integrity levels, as follows:
    - **0–3** User Integrity
    - **4** Operating System Services (OSS) Integrity (i.e., reserved for OSS untrusted applications).
    - **5** Operator Integrity
    - **6** *(not predefined)*
    - **7** Administrator Integrity
  - Integrity categories (16 nonhierarchical)

Given two labels (levels and categories), the first is considered to "dominate" the second if the hierarchical level of the first is greater than or equal to that of the second, and if the category set of the first is a superset of the second.[1]

MAC labels are stored internally in the following format:

[1] This comparison rule holds for both sensitivity and integrity labels; the "duality" of integrity is handled by swapping the order of comparison (object dominates subject, as opposed to subject dominates object).

Sensitivity Level	An 8-bit byte, constrained to values 0 through 15.
Sensitivity Categories	64 bits, where setting a bit indicates possession of a particular category.
Integrity Level	An 8-bit byte, constrained to values 0 through 7.
Integrity Categories	16 bits, where setting a bit indicates possession of a particular category.

6.2.1.2 Policy Rules

The MAC policy enforced using these labels is a combination of the policies defined by the Bell and LaPadula model [4] and the strict Biba model [5]. This policy uses the following access rules:

- Simple Security Policy

  A subject may access an object for reading if the sensitivity label in the current MAC label of the subject dominates the sensitivity label of the object.

- Simple Integrity Policy

  A subject may access an object for reading if the integrity label of the object dominates the integrity label in the current MAC label of the subject.

- Security *-Policy

  A subject may access an object for writing if the sensitivity label of the object dominates the sensitivity label in the current MAC label of the subject.

- Integrity *-Policy

  A subject may access an object for writing if the integrity label in the current MAC label of the subject dominates the integrity label of the object.

Since the policy used is a combined policy, both the simple security and integrity policies must be satisfied to access an object for reading. Similarly, to access an object for writing, both the security and integrity *-policies must be satisfied. Control of creation and deletion of hierarchically structured objects, such as those in the file system, are based upon the ability to write the directory containing the object.

An earlier version of the Bell and LaPadula model than [4] also required that the level of the object should not change ("the Tranquillity Principle"). [4] proves that it is possible to change the level of an object without violation of the policy rules just described. This requires accesses by a subject to an object be checked and possibly revoked when an object level changes. XTS-300 allows object levels to change (consistent with policy), revalidating access whenever this happens.

6.2.2 Discretionary Access Control Policy

In contrast to the MAC policy, the DAC policy grants access based on a relationship between a named user and a named object. All subjects have associated with them (in the Active Process Table Entry (APTE) for the process) the identity of the user and group on whose behalf the subject operates. This association is established at the time of process initiation, and can be changed only by the TCB.

In a manner similar to UNIX, STOP 5.2.E supports the concept of real and effective user identification. Access checks are done based on the current effective user and group identification. This identification may be changed. The real user and group IDs are retained, independent of changes to the effective user, for auditing purposes. A separate gate, usable for this purpose only by trusted code, must be used to change the real user/group IDs.

The potential access allowed[2] to a subject, operating on behalf of a named user/group, is specified by the Access Control List (ACL). In the STOP 5.2.E system, ACLs are stored along with the object subtype. (see Section 6.3.2) information. Each ACL contains the following information:

- Owning User ID of the object, and the allowed access modes of that user. For a process, this user ID is interpreted as the effective user ID.
- Owning Group ID of the object, and the allowed access modes of that group. For a process, this group ID is interpreted as the effective group ID.
- A list of up to seven (see Section 9.7, page 136) other users or groups,[3] and their allowed access modes.
- Allowed access modes for any other user or group that has not already been covered.

When an object is accessed, the ACL for the object is examined to find the first entry that matches the effective user or group of the subject that issued the access request. This search is equivalent to the following. The specific algorithm used in STOP 5.2.E is shown in Figure 6.2.

1. The owning user is compared with the effective user. If they match, the permissions for the owning user are used.
2. If the owning user does not match the effective user, the ACL is examined to determine if there are any ACL entries that match the effective user requesting access. If an entry is found that matches (the first one found is used), the permissions associated with that ACL entry are used.
3. If no match is found for the effective user in the ACL, the owning group is then compared against the effective group. If this comparison is successful, the permissions for the owning group are used.
4. If the owning group does not match the effective group, the ACL is examined to find the first entry (if any) that matches the effective group. If such an entry is found, the permissions associated with that entry are used.
5. If no match is found for the effective group in the ACL, the "others" permissions are used.

Once a match is found, the permissions are examined to determine if the requested mode of access is one of the allowed access modes for that subject.

There are three allowed access mode bits in each ACL entry; any combination is syntactically valid (although it may have no semantic meaning). These bits are:

read If this bit is set, the user or group is allowed read access to the object (if allowed by the MAC policy).

[2] The term "potential" is used because the ACL does not specify the current access, only the allowable access. The potential access becomes an actual mode of access when the subject opens the object in a particular access mode.

[3] A bit is used to indicate whether the given ID refers to a group ID or a user ID.

write If this bit is set, the user or group is allowed write access to the object (if allowed by the MAC policy).

execute If this bit is set, the user or group is allowed "execute" access to the object (if read access is allowed by the MAC policy). This bit may be ignored or be meaningless for certain types of objects (for example: devices, named First-In First-Out (FIFOs) processes). For directory file system objects, this bit is not interpreted as "execute," but as "search."

If the particular bit for that type of access is not set, the corresponding user or group is denied that mode of access. Hence, to deny access to a given user or group, the bits for all types of access would not be set.

6.2.3 Enforcement of Policy

Although the basic policy enforced is the same for all subjects and objects, there are differences in the specific mechanisms used to provide that enforcement. This section presents these specifics, including, for each type of object, discussions on: where policy is enforced for the object in the system, how a user can change the attributes of the object, and what happens as a result of changes to the attributes of the object.

Proprietary material removed.

6.3 Additional Supporting Protection Mechanisms

In addition to the TCB mechanisms that provide enforcement of the MAC and DAC policies, STOP 5.2.E provides four additional protection mechanisms that are not directly policy related. These mechanisms provide additional TCB protection, support of least privilege, and a controlled means of bypassing security policy.

6.3.1 Descriptor Privilege Level

This is a hardware mechanism used to restrict access to segments in memory, thus providing TCB self-protection. This is described in more detail in Section 3.4.3.5, page 25.

6.3.2 Subtypes

Another supporting TCB protection mechanism is the subtype mechanism. Subtypes are like tokens in a capability-based system; to access an object, a subject must possess the object subtype for the object.

This mechanism is used by the kernel to restrict access to objects. The system supports subtypes for processes, segments, and devices. The primary use of subtypes is to provide control over the trusted path – when the Secure Attention Key (SAK) is pressed, the Server changes the subtype of the terminal to prevent any untrusted processes from accessing it. Other uses of subtypes in the system are as follows:

- The trusted program fsm uses segment subtypes to provide exclusive file access. When fsm starts, it changes the subtype of all the segments that support the object to a reserved value; thus, any untrusted processes having the object opened will lose access. This use of subtypes also provides an interlock for multiple fsms operating on the same file system object. At the completion of fsm, the original subtype is restored.
- Subtypes are used to protect audit files and trusted databases. Only trusted programs are given the appropriate subtypes necessary to access the files and databases.

Subtypes are stored with the ACLs and are part of SBTE, DBTE, and the APTE. Internally, subtypes are stored as 16-bit words. The value of zero is the default – all subjects have access to objects with a subtype of zero. Each subject has an accessible subtype list consisting of up to five subtypes for each object type (segments, devices, and processes). Only the default subtype is used for process objects.

For devices, terminals are assigned a subtype consisting of the index into the terminal configuration data base plus the value of the process family currently attached to the terminal. Each process family is given only the appropriate subtype. With this mechanism, it is assured that only one process family can access the terminal; the terminal subtype is set based on the reattach command. When the trusted path is invoked, the process family portion of the subtype is set to zero, which assures that only trusted software can access the terminal.

For segments (files), only two additional subtypes are used: *TRUSTED_DATA_SUBTYPE* and *AUDIT_DATA_SUBTYPE*. The audit subtype is used to restrict access to audit data (both the audit files and the controlling audit database) to trusted software.[4] The trusted data subtype is used to restrict access to various trusted databases such as the user access database containing passwords to trusted software.

Subtypes are checked by the kernel on every operation on the object. They are initially checked when the particular object is first introduced into the subject's address space (i.e., at the time of mapping) or when the object is accessed (if mapping is not required). When an object subtype is changed, the descriptor is invalidated, which forces the subtype to be rechecked. When the subtype list for a process is changed for a particular category of subtype, all access for that category is rechecked. Accesses where the subtype has not changed can be viewed as having the subtype checked implicitly, because revalidation of the subtype does not occur.

Checking of subtypes is done by comparing the appropriate subject's subtype list (contained in the APTE) to the subtype of the object. In order for the subject to have any form of access to the object, the subtype of the object must be on the subtype list. The ability to add subtypes to the subtype list of a subject is controlled by the privilege mechanism.

6.3.3 Capabilities

The third supporting TCB protection mechanism is the capability mechanism, which provides a way for the system to restrict the ability of users to use commands, thus enforcing a secondary layer of least privilege on top of that provided by the hierarchical integrity level. Capabilities are a mechanism whereby the commands and possible actions available to a user from the Secure Server and other trusted software can be restricted.

[4] Although the audit files are protected by the audit subtype, fsm can be used to manipulate audit files by administrators running at maximum sensitivity and maximum integrity. With fsm, audit files can be displayed, renamed, and deleted, but not copied.

It should be noted that support for the capability mechanism is localized to the Secure Server and the trusted programs called by the Secure Server. Auditing of the use of capabilities is not done directly; it is indirect through the auditing of the actions taken by the trusted programs.

Capabilities are defined for a user by the administrator through the ua_edit program. The system understands the following capabilities:

DOWNGRADE ALLOWED	This capability gives a user the ability to set the MAC label of a file system object to a lower value.
UPGRADE ALLOWED	This capability gives a user the ability to set the MAC label of a file system object to a higher value. The two capabilities mentioned above (DOWNGRADE ALLOWED and UPGRADE ALLOWED) are used by fsm, which is a trusted program. Fsm raises the *SIMPLE_SECURITY_EXEMPT* privilege when modifying the attributes of a file; it controls the ability to upgrade or downgrade the MAC label of a file through the use of the DOWNGRADE ALLOWED and UPGRADE ALLOWED capabilities. In addition to the capabilities, the user issuing the request must be the owner of the file system object (unless the user's integrity level is administrator or higher) and must have a current MAC label equal to the MAC label of the file system object. Note how this differs from the checks made when copying files, which is a service provided by the untrusted (and unprivileged) CASS.
VIEWING OPTIONAL	This capability gives a user the ability to bypass viewing the contents of a file when downgrading it.
CUP ALLOWED	This capability gives a user the ability to use the cup program to change the user's password (or another's password, if the user has an integrity level of administrator or greater).
DISCONNECT ALLOWED	This capability gives a user the ability to disconnect process families from the current session and let the process families operate in the background. It also controls the ability to run after logout.
KILL ALLOWED	This capability gives a user the ability to use the kill or ikill commands to send a kill signal to processes associated with the current session.
RUN ALLOWED	This capability gives a user the ability to use the run command to invoke the user's default program. It also controls the ability to use the chd and ccp commands.
SG ALLOWED	This capability gives a user the ability to use the sg command to change the user's current group.
SL ALLOWED	This capability gives a user the ability to use the sl command to change the label of the current session. It also controls the ability of a user to change the user's default label (cdl command).

SHUTDOWN ALLOWED	This capability gives a user the ability to use the shutdown command to bring the system to an orderly halt. It also controls the ability to generate a system dump via the dump command.
UNMARKED PRINT ALLOWED	This capability gives a user the ability to generate printouts that do not have sensitivity labels at the top and bottom of each page. It also allows certain printer escape sequences to be sent directly to the printer. This capability does not suppress the labeling of the banner pages.
MULTIPLE LOGIN ALLOWED	This capability gives a user the ability to be simultaneously logged in on multiple terminals.

Capabilities are checked by Trusted Software.

6.3.4 Privileges

The last supporting TCB protection mechanism does not actually provide protection; rather, it provides a controlled mechanism whereby a process operating on behalf of a user can be authorized to bypass the system security policy in a selected fashion. This is done by associating with each form of policy bypass a specific privilege that must be possessed by the process to exploit the bypass.

Every executable file in the file system has associated with it a maximum privilege set that represents the maximum set of privileges that an instance of that program may have at any point in its lifetime. This maximum privilege set may not be changed by untrusted users; a user with an integrity level of administrator must use the tp_edit program (see Section 4.7.3.3.10, page 77) to change it. Thus, only trusted programs have privilege.

Every active process on the system has associated with it (in the APTE) both a maximum privilege set and an effective privilege set. When a process is loaded, the integrity level of the program file is examined. If it is a trusted program with an integrity level of operator or above, the maximum privilege is obtained from the control segment of the executable file as described above. If it is not a trusted program, the maximum privilege set is set to the empty set, unless the invoking process possesses the *TRUSTED_PARENT_EXEMPT* privilege. The effective privilege set for all processes starts out empty. As a program executes, it may use the kernel gate `set_process_status`, through the `set_privilege` or `add_privilege` function, to change its effective privilege set; however, this set must always be a subset of the maximum privilege set.

A process may also change its maximum privilege set dynamically through the `set_process_status` gate. To do this, however, it must currently possess the *MODIFY_PRIVILEGE* privilege. The trusted program editor, tp_edit, does not allow the administrator to assign this privilege to any trusted program; thus, it can be present only on programs configured in by the vendor.

The following privileges are understood on the STOP 5.2.E system:

- *MODIFY_PRIVILEGE*. Allows a process to modify its maximum privilege set.
- *SET_LEVEL*. Allows a process to change the MAC label of an object.
- *UPGRADE_LEVEL*. Allows a process to upgrade the MAC label on an object.

- *SET_DISCRETIONARY_ACCESS*. Allows a process to change the access control list of an object if it is not the owner of the object.
- *SET_OWNER_GROUP*. Allows a process to change the access control list of an object, the other mode bits (i.e., setuid/setgid) of an object, or the owning user/group of the object, even if it is not the owner of the object.
- *SET_PROCESS_ATTRIBUTES*. Allows a process to set its clearance label and process family.
- *SET_SUBTYPE_ACCESS*. Allows a process to change the current subtype of an object.
- *TERMINAL_LOCK*. Allows a process to retain control of the terminal when a secure attention key is pressed (see Section 4.7, page 62).
- *DEVICE_CONTROL_EXEMPT*. Allows a process to perform primitive hardware control functions on a device (e.g., loading the controller firmware).
- *SIMPLE_SECURITY_EXEMPT*. Allows a process to bypass the simple security property – i.e., it can read objects at a sensitivity label that dominates the process' sensitivity label.
- *SECURITY_STAR_PROPERTY_EXEMPT*. Allows a process to bypass the security *-property – i.e., it can write objects with sensitivity labels dominated by the process' sensitivity label.
- *SIMPLE_INTEGRITY_EXEMPT*. Allows a process to bypass the simple integrity property – i.e., it can read objects at an integrity label dominated by the process' integrity label.
- *INTEGRITY_STAR_PROPERTY_EXEMPT*. Allows a process to bypass the integrity *-property – i.e., it can write objects with integrity labels that dominate the process' integrity label.
- *DISCRETIONARY_ACCESS_EXEMPT*. Allows a process to bypass the discretionary access and subtype policies.
- *TRUSTED_PARENT_EXEMPT*. Allows a process with an integrity level below operator to load trusted processes with privileges. When the privilege is present at the time of loading, the privilege bits of the new process are not zeroed when the creating process is untrusted. It is not currently used by the STOP 5.2.E.

6.4 Identification and Authentication

STOP 5.2.E requires all users to identify and authenticate themselves before they are allowed to access system resources. Users identify themselves by entering a unique username, and authenticate their identity by entering a password. The username and password are initially assigned by the site system administrator. A user is identified as a system administrator by the current MAC label associated with that user. If the label includes administrator integrity, then that user may execute system administrator commands.

6.4.1 Trusted Path

A trusted path is established by pressing the SAK which is the <BREAK> key. The trusted path cannot be initiated by a program or without the user's knowledge because the signal must be generated by hardware.

All user actions requiring the protection of a distinct user-to-TCB communication utilize the trusted path. The trusted path is utilized for login processing, logout, process control, changing passwords, changing groups, changing home directory, changing access attributes of a file, changing the security level of the working user level and system devices, displaying session or system status, outputting a file to a printer or terminal, and mounting or unmounting a file system. It is also used for operator and administrator commands. The TCB does not initiate any communication to the user via the trusted path.

The TCB does not support trusted path across networks or for pseudo-terminals. In addition, unprivileged processes cannot establish a login pseudo-terminal.

6.4.2 The Login Sequence

Proprietary material removed.

6.4.3 Password Aging

There is a password expiration date and a password lifetime date that is associated with all user IDs on the system. The value for each is stored in the Trusted Information database and applies to all users on the system. These values are site configurable; however, the default values are 20 weeks for the expiration date and 26 weeks for the lifetime date.

When the expiration date is reached, the user is informed and must then update the password in order to log in to the system. Users must have the CUP ALLOWED capability in order to change their passwords. Otherwise, the system administrator must change a user's password. If the password is not updated by the time the password lifetime date is reached, the password is invalid and the user is locked out of the system. A locked user can log in only after an administrator has changed the user's password. An exception is made to allow an administrator to log in at the system console even if the user ID for the system administrator is locked.

6.4.4 Failed Logins

A terminal is locked once a site-specified number of failed login attempts occurs. The default value is five and is stored in the Trusted Information database. A terminal lock is implemented by the system ignoring the SAK. Whenever a terminal is locked, an audit record is generated and a message is sent to the system console. The terminal will remain locked until its lockout interval elapses or until the system administrator issues a cancel terminal lockout command, ctl, to clear the lock.

A terminal's lockout interval is obtained from the Terminal Configuration database entry for that particular terminal (if the value is nonzero) or from the default lockout time in the Trusted Information database where the default value is 60 seconds. Each terminal may have its own lockout time defined in the Terminal Configuration database. When a terminal is locked, it is locked to all users.

6.4.5 Session Control

If a default command processor is specified for the user in the User Access Information database, and the user's integrity level is below OSS integrity, the successful completion of the login procedure results in the automatic execution of that command processor as an untrusted process. Successful loading of the command processor program results in a "leaving trusted environment" message on the user's terminal.

Once the user is logged in, pressing the SAK causes the current process to become detached and a trusted path to be established by the TCB. The Secure Server displays the current MAC label and family process id. A prompt for the next command is displayed. At this point, the user can issue Trusted Commands, (see Section 4.7.3, page 64). The SAK must be pressed for each Trusted Command that is to be issued. The reattach command is used to reconnect a terminal to a process that was detached by the use of the SAK. The session is terminated when the logout command is invoked.

6.4.6 Logout

The logout command is used to terminate a terminal session. All active processes that are associated with this terminal session are killed. Any process from which the user has disconnected via the disconnect command continue to run. The ownership of the terminal is reset to indicate it is not logged-in. An audit record is generated for the logout event.

6.5 Set User ID Protection

Proprietary material removed.

6.6 Audit

Proprietary material removed.

6.6.1 Audit Events

Audit events are generated by Trusted Software, TCB System Services, and the kernel. The following audit events are generated by Trusted Software:

- Print request issued with no markings
- fsm request failed
- Trusted editor service performed
- Change default level command issued
- ctl command issued

- cup command issued
- Login attempted
- logout command issued
- sg command issued
- sl command issued
- st command issued
- startup command issued
- shutdown command issued
- Administrator command issued
- Operator command issued
- Device start error.

For the administrator command event and the operator command event, the audit record includes the particular command that was issued. The trusted editor audit event includes the name of the editor that was invoked, as well as the service that was performed.

TSS generates audit messages for the following security-related events:

- Discretionary access denials to file system objects
- Opens and closes of file system objects and sockets
- Creates and deletes of file system objects
- Ownership and access changes of file system objects
- Installation/removal of set user ID programs
- Program loader failures.
- Adding and removing links
- Mounting and unmounting of file systems
- Failure to successfully bind a socket
- Connect or accept failures for sockets
- Sendto and recvfrom operations for sockets
- Registering and attaching a socket
- Inbound connect failure for network
- ICMP redirect for network

The Security Kernel generates audit messages for the following security-related events:

- Device/process creation
- Device/process deletion
- Device map
- Device unmap
- Disk error
- IPC message sent
- Process duplication
- Unmounting of a busy file system
- Any operation that changes the segment/device/process owner, security level, integrity level, or process privileges/accessible subtypes
- The exhaustion of the following resources (for covert channel purposes):
  - Available processes
  - Segment space within a file system
- Any access attempt denied because of security or integrity violations
- Any access attempt denied because of discretionary access violations
- Any access attempt denied because of subtype access violations.

6.6.2 Contents of an Audit Record

Each audit record is composed of a header and a data section. The header contains the following information:

- Size of the audit record
- Type of event being audited
- Time the audit record is generated
- Process ID of the process causing the audit event
- MAC label of the process
- Effective privileges of the process
- Real user ID
- Real group ID.

The data portion of the audit record contains data pertinent to the particular audit event including, but not limited to, such things as the device ID, MAC labels including both the new label and the old label in the case of a change, file system ID, segment name, and privilege set.

6.6.3 Audit Preselection

The param_edit command allows preselection of auditing by event and by user by updating an event list and user list in the Audit Information database. In addition, the auditing mechanism is implemented such that a minimum MAC label may be specified below which audit messages are not generated for object creation, deletion, and access. This MAC label value is also stored in the Audit Information database. This database is protected using file subtypes where access is limited to param_edit and the kernel.

The kernel determines whether to generate an audit record based on the state of the event list (that event is enabled) and the state of the user list (the user ID/real process user ID, is enabled). The Secure Server forces login and logout to be audited independently of the state of the user list.

6.6.4 Audit Post-Selection

The Trusted Software audit command formats the raw audit data to allow management of the audit files. The audit command accepts one of five commands:

switch	switch the current audit files in order to accumulate audit data in a new file
remove	remove audit files
files	list the existing audit files on the system by name, creation time, and file size
display	display the existing audit data
exit	exit the audit command

The display command is used to view the audit data through the use of the following subcommands:

print	displays the selected audit files on the terminal or on a specified printer
reset	resets the criteria that have already been defined for selection.
select	selects audit records from the specified audit files according to the criteria specified in this subcommand.

The valid selection criteria are audit event type, start date and time, stop date and time, process id, user id, group id, device ID, segment ID, trusted editor command name, trusted editor request, file name, object level, and audit records generated for a specified range of MAC labels.

show	displays the current selection criteria
quit	exit the display function.

The audit command is restricted to users whose integrity level is at least Operator and whose security level is at system maximum. Additionally, to display or delete audit information, the user must be of at least the Administrator integrity level and at the system maximum security level.

6.6.5 Monitoring Security Auditable Events

STOP 5.2.E monitors as imminent security violations the accumulation of repeated failures of login attempts. For failed login attempts, an audit record is generated for each attempt. After repeated failed login attempts, a message is sent to the system console and the terminal is locked out (see Section 6.4.4, page 94).

6.7 Object Reuse

Objects examined under the XTS-300 include the following: segment-based objects, directory entries, and hardware objects. Segment-based objects include segments, processes, files, directories, device special files and named FIFOs. Directory entries contain the file name and the associated segment unique ID (uid). Hardware objects include Central Processing Unit (CPU) registers, controller boards or user devices.

6.7.1 Segments

Segment growth occurs only at segment creation time in units of pages. The Segment Manager makes sure that a segment whose pages will not be filled with information from disk storage will be cleared upon allocation by the Memory Manager `retrieve_page` routine. All segment-based objects are created or grown through calls to common routines in the Segment Manager and Memory Manager. When a segment shrinks, the TCB clears residual data in partial pages.

6.7.2 CPU Registers

The Kernel never propagates the values of system registers out to a process, even indirectly. Unprivileged software can read some of the system registers, and the privileged bits in the flags register, but only the CR2, LDTR and TR registers are modifiable (indirectly) by a process. However, LDTR and TR always point to structures for the current process, so no information can be seen from a previous process.

The Intel Pentium II/III CR2, CR3, and CR4 can not be read outside Ring 0. The general purpose, segment, instruction pointer, and flags registers are always completely saved and restored across hardware task switches. The Kernel uses these tasks to implement processes, so there is no way for a process to see data left over in one of these registers from a previous process. The MSRs can not be read, even indirectly, by intrusted software, except for the time stamp. Other untrusted code can not modify this register (even indirectly), however, so there is no object reuse issue. The APIC registers can not be read or written by untrusted software. This includes the APIC time stamp counter (TSC) which is actually disabled by setting the CR4 register disallowing any access to it outside privilege level 0.

The instruction prefetch queue is flushed during task switches. There is also no way for a task to read a portion of the prefetch queue not meant for the process. There is no way for a task to read the branch target buffers on the Intel Pentium II/III (also they are flushed during a task switch because CR3 will be reloaded). The Intel Pentium II/III has additional internal registers and buffers to support dynamic execution, but the argument is the same.

Although the TS bit in CR0 can be read by untrusted code using the SMSW instruction, the XTS-300, ensures

that the TS bit remains in the same state on dispatch that it was in when the process last relinquished the processor. In this way there is no covert channel involving this bit.

All entries into the TSS are cleared when allocated to a new process. Furthermore, the TSS is always initialized to a known state during the load of the system. From that point on, hardware interrupt handling will save and restore registers properly.

6.7.3 I/O Device Registers

Except for I/O registers associated with the video card, all access to I/O registers is limited to the kernel. There is no access outside the kernel to any of the other I/O registers. Access to the video I/O registers is discussed below in the section on Terminals driver. The software drivers only support the ability to return the value of the status registers. A user cannot modify the status register in any way. A user can examine the value of a status register. In essence, the user cannot place any arbitrary value in the status registers.

There are eight classes of devices available to users via the TSS I/O interface. These include terminals, disks or diskettes, magnetic tapes, PC card, data transfer cartridges, host adapters, sockets and networks. The Small Computer Systems Interface (SCSI) controller which supports both disk and tape is included in this discussion. Each of these will be examined from the point of object reuse.

Disks	The contents of the 16-bit status register is set to a specific value by the hardware when the disk is opened by a user. This is accomplished by issuing a SEEK to cylinder zero to the disk drive.
Tapes	Upon powerup, the drive performs a series of self-tests. An early failure will result in the drive disconnecting itself from the SCSI bus while a later failure will result in an error being returned to the Host Adaptor. Registers and internal memory (data buffer) are initialized and the drive is placed in a known state. The tape is repositioned when the tape drive is opened by a user. The contents of the 32-bit status register is reset to a specific value by the hardware.
Printers	For serial printers, the contents of the 32-bit status register is set to a specific value by the hardware when the printer is opened by a user. This is accomplished by issuing a FORM FEED to the printer. For parallel printers, a FORM FEED is also sent to the printer so that the status register is set to a specific value.
Networks	Upon powerup, the Ethernet adaptor is placed in a known state. Both the low-level buffer addresses and the control registers that are mapped into the XTS-300 I/O address space are inaccessible outside of the kernel. Normal I/O is accomplished through command blocks which includes status information. All command blocks are flushed (zeroed) during an unmap of the device.
Sockets	Sockets reside in the shared memory segment that is created and controlled by the TCP/IP daemon. This shared memory segment is used for communication between sockets and the TCP/IP daemon. The shared memory segment contains a segment header and socket structures needed for the socket implementation.

The shared memory segment header contains controlling information for socket management. This header is initialized by the TCP/IP Daemon when it creates the segment. This header is meant to pass information to all processes that use this segment. There are no user interfaces to provide the outer rings with information contained within the segment header.

The shared memory segment is only accessible within Ring 1. When a TCP/IP Daemon terminates, it removes the old shared memory. Also, when a TCP/IP Daemon starts, it must create a new shared memory segment for communication with processes using sockets. This prevents any previous socket/network activity (using the old segment) from interfering with the new segment. All new processes attempting to use sockets will use the new segment and existing processes will never have access to the new segment.

The socket structure with controlling information is properly cleared on allocate. In addition, the mbuf header except for the length field is initialized on allocate. The calling routine then sets the length field based on the data that is being used. Data outside the length field is not referenced. This is similar to the `get_io_buffer` mechanism used by other TSS device drives.

Terminals

The user ensures that internal and external storage of a terminal is cleared between sessions. The console is a special case of a terminal and should be distinguished even further by the fact that an outer ring can directly manipulate the console video registers/memory in support of graphic based applications. However, the following rules are enforced:

Before a program is given direct access to the video adapter, the following steps are performed:

- All of video memory is cleared (after saving the portions of video memory that are needed to produce the text display).
- Registers that are modifiable under program direction from within text mode are set to "constant" values. These registers define the cursor type and the cursor position on the screen.

When the program ceases to have direct access to the video adapter, all video registers are restored and the portions of video memory needed for text mode are restored.

These actions have the following implications for object reuse:

- A graphics mode program never sees any left-over video memory contents, since all of video memory is cleared.
- For video registers that are not modified during text mode operation, a graphics mode program only sees the register contents that were loaded when the system was booted, since registers are restored after each use by a graphics mode program.
- For video registers that are modified during text mode operation, a graphics mode program only sees the constant values that were loaded when graphics mode was entered.

Host Adapter	Upon powerup, the host adapter's registers and internal memory (data buffer) are initialized and it is placed in a known state. Note that the host adapter itself has already been included in the discussion on SCSI peripherals such as disk and tape. Any I/O to the Host Adapter device itself never causes I/O to a SCSI device or adapter. Rather, static data that is initialized at system initialization time (such as the existence and size of a SCSI peripheral) is always returned.
PC Card	There are no hardware data structures or registers that are accessible to untrusted software. However, status information is available to untrusted software via the MODE_SENSE command and request sense data returned after a read or write error. These values properly reflect the status/attributes of the device at the time of the call.
DTCR	This device is similar to the PC Card device in that no hardware structures or registers are accessible to untrusted software. Status information is available to untrusted software via the REQUEST_SENSE, MODE_SENSE, READ_DEFECT_DATA, and RECEIVE_DIAGNOSTICS commands. These fields cannot intentionally be set by a process. Like the PC Card, these values properly reflect the status/attributes of the device and do not contain process data.

6.7.4 Directory Entries

When a file is deleted under XTS-300, both the segment uid of the control segment for the file and the name of the file is deleted, leaving no residual information in the directory (this is a variation from standard UNIX behavior).

6.7.5 Removable Media

XTS-300 does not support electronic labels on removable media, such as tapes and diskettes. Object reuse on removable media is accomplished by a combination of electronic and procedural controls. The procedures for handling cartridge tapes are described in the Trusted Facility Manual (TFM). The following is a summarization of the procedure for cartridge tapes and diskettes. The cartridge tape procedure can be applied CD-ROMS, PC card, and data transfer cartridges.

6.7.5.1 Tapes

A request is made for a tape drive to the operator by the user via a telephone call or some other means of communication external to the system. The user provides the operator with a user ID and a desired MAC level for the device. After checking whether there is a cartridge tape already mounted on the drive,[5] the operator issues the set_device_access sda command. Using the sda command, the operator will set the level of the device to that specified by the user and will also set the ACL of the device such that the requesting user has exclusive access to the drive.

[5] The operator should eject the tape if it is already present

Once the operator has completed the setup, the user issues a programmatic `mount_tape_or_disk` request, that includes the user's ID, the MAC level of the requesting process, the volume (cartridge tape) name (if any), and the mode of access desired, read and/or write. The TCB than validates access based on the level of the requesting program and the level of the device. If access is granted, a mount request will appear on the system console.

If no volume name is provided, the TCB requests the operator to mount a scratch tape. In the case in which a specific volume (cartridge tape) is requested, the TFM recommends that the operator again issue the sda command to set the MAC level and DAC ACLs for the tape drive to that specified on the external label of the cartridge tape. The tape is then placed on the drive by the operator. The TCB will revalidate access to the device after the operator responds to the mount message.

If any cartridge tape is currently mounted and ready to use, and a request to mount a specific volume is made, the TCB will dismount the current cartridge tape. The operator must then use the sda command to set the security attributes of the tape drive based on the external label of the requested cartridge tape as described above.

It is not necessary for a user program to unload a tape upon program termination. A tape may be left loaded for subsequent use by other programs. The TCB will validate access to the tape for each subsequent request so no security compromise is possible. If a tape is already loaded in the drive and a scratch tape is requested, the operation of the user's program will resume immediately. If the user's program provides a volume name and there is a tape already loaded in the tape drive, the tape on the drive will be unloaded. The TFM states that the operator should set the access attributes of the drive based on the external tape label associated with the new tape request. Removable media may be declassified only after undergoing a site-approved degaussing procedure.

6.7.5.2 Diskettes

Diskettes may be used in two ways: as user-owned media or as system media. In the case of use as a system diskette, diskettes are treated as are system disks. Access to system diskettes is controlled through the sdc, mount, and unmount Trusted Software commands. If diskettes are treated as user-owned media, they follow the same procedures as for cartridge tape handling.

Diskettes may be used as a boot media under XTS-300 but only by someone who has access to the system console.

6.8 Usage of Tapes under XTS-300

XTS-300 does not support electronic labels on tape cartridges. The TFM describes a procedure for mounting tapes.

The following is a sequence of events which transpire when a user wants to use a tape cartridge.

The initial request to use a tape drive is made externally to the system, at a face-to-face meeting between the user and operator or by a telephone call. The user provides a user ID and a MAC level. The operator first checks that there is no tape cartridge mounted on the tape drive and subsequently issues the set_device_access (sda) command. This sda command will set the MAC level of the tape drive device to that which the user

requested, and the DAC (ACL entry) such that the requesting user has exclusive access to the tape drive (first ACL entry is set Read/Write for the user's ID). Since the system devices are owned by "system," untrusted users cannot modify the access to these devices.

When this is completed, the user issues a programmatic `open_disk_or_tape` request,[6] which will include the user's ID, the MAC level of the requesting process, the volume (tape cartridge) name (if any), and whether the tape cartridge will be modified. Access to the tape drive device will be validated based on the initial `sda` command. If access is granted, the mount request will be relayed to the operator at the system console.

If no volume name is provided, the TCB requests the operator to mount a scratch tape.

If a specific volume (tape cartridge) is requested, the operator should again use the `sda` command to set the MAC level and DAC ACLs for the tape drive device based on the information provided on the external label of the tape cartridge. The tape cartridge is placed on the drive at the end-of-tape mark and readied. The TCB will then re-validate access to the tape drive device based on the new security attributes provided.

When actually physically mounting the tape cartridge, the operator can enable or disable write protection on the cartridge depending on the information provided in the mount request. If write mode was requested and the write enable mechanism is disabled, the TCB rewinds and dismounts the tape cartridge and makes a request that the operator write enable it. Subsequent checks are made until the condition is satisfied.

Before the user program terminates, it should close and dismount the tape cartridge. When a writable tape has been dismounted, the operator must ensure that the tape cartridge has an external label which includes the current MAC level of the tape drive. The current DAC information of the tape drive should also be on the external label.

If any tape cartridge is currently mounted and ready to use, and a request to mount a specific volume is made, the TCB will dismount the current tape cartridge. The operator must then use the `sda` command to set the security attributes of the tape drive based on the external label of the requested tape cartridge as described above.

The TFM contains additional specific guidance on how to handle tape volumes.

If the tape cartridge is write-protected, then it is permissible to mount it if the MAC level on the external label of the tape cartridge is lower than that of the requesting program.

On dismounting a tape cartridge, the operator should always write the MAC level of the tape drive device on the external label of the tape cartridge.

All scratch tape cartridges must have an external label indicating the highest MAC level at which it was used. Scratch tapes are assumed to allow universal discretionary access.

Magnetic media may be declassified only after undergoing a site-approved degaussing procedure.

[6] The TSS `open_device` gate may be called directly, passing in the same information.

Chapter 7

Assurances

7.1 TCB Layering

Layering principles are evident on two levels in XTS-300. One is on a domain level. Starting with the Kernel domain, the domain of greatest privilege, the ability to control access to Trusted Computing Base (TCB) objects diminishes from the Kernel domain (Ring 0) to the TCB System Services (TSS) domain (Ring 1) to the Operating System Services (OSS) domain (Ring 2) to the User domain (Ring 3). In addition, there is layering within the Kernel domain and the TSS domain.

The Kernel domain is the primary area of security policy enforcement. The kernel enforces Mandatory Access Control (MAC) policy, performs as a reference monitor on all TCB objects such as segments and devices, performs physical I/O on all devices, performs auditing on a segment level, and performs Ring 0 initialization including creating a process execution environment for kernel and TSS processes. The TSS domain provides the file system hierarchy (i.e., the kernel is only aware of a flat file system made up of segments), enforces Discretionary Access Control (DAC) policy, implements the network protocol stack and provides high-level services for user I/O. In a TCB process, the Trusted Software running in the Operating System Services domain provides the user interface (i.e., trusted path), manages process family relationships, performs all system initialization with the exception of Ring 0 initialization, manages all the TCB databases including user profile information, and enables privileges for software which performs security-relevant functions.

Within the kernel and TSS there is an internal layered structure. Functions at higher levels generally rely on services provided by lower-level functions. Those functions at lower levels generally rely on less functionality to execute.

Layering violations within the kernel were permitted in cases of detection of unrecoverable errors or during interrupt processing. There are ten additional occurrences of kernel layering violations that do not pertain to error or interrupt processing. It appears that no data abstraction principles have been compromised within the layering violations. In addition, it appears that none of them lead to any recursive calls.

7.2 Covert Channel Analysis

The vendor has analyzed XTS-300 running the STOP 5.2.E system for sources of covert channels [20]. The methodology for determining the shared resources is described below.

7.2.1 Shared Resource Analysis

The vendor's analysis of the shared resources was conducted by a thorough review of all resources. In order to bound this analysis, the vendor excluded those TCB components that do not provide a program interface.

The only sources of shared resources that can be used as covert channels are the security kernel and trusted processes exempt from the mandatory security policy. The security kernel is the primary source of covert channels since the kernel does not enforce policy on itself when it accesses internal resources; it is exempt from policy. Similarly, trusted processes, which are exempt from the mandatory security policy and can be invoked by untrusted software, may also be a source of covert channels since they, like the kernel, are exempt from policy. Trusted processes that can be invoked only directly by a user (not by a program) cannot be used as a source of covert channels. Any "object sharing" within the non-kernel domains of an unprivileged process cannot be used as a source for covert channels because all access is mediated by the kernel; all such object accesses comply with the system's mandatory security policy.

The guidelines described above were used to determine the following five basic categories of shared resources:

1. Global variables that exist within the kernel upon the completion of startup
2. Global variables that are dynamically allocated and deallocated by the kernel during execution
3. Process-local variables that are accessed by the kernel on behalf of another process
4. Variables that are accessed by TCB daemon processes exempt from mandatory policy
5. All hardware components.

The first three categories are data variables under control of the kernel. The vendor created lists of these global resources by reviewing the kernel design document and checking against the kernel source code. The fourth category, those resources that are accessed by the TCB daemon processes, was compiled from the Trusted Software and TSS design documents. The last category includes all hardware components because all hardware is sharable.

A second phase of analysis was then conducted which consisted of examining each resource obtained in the above process to see if it could be used in any manner such that one process could modify it, and another process could examine it, even if either modification or examination required indirect intervening steps. The resources that resulted were then analyzed to determine if there were any constraints on their use (such as restriction of modification or examination to privileged processes), causing them to be eliminated from further consideration.

The remaining resources were then examined individually, and a covert channel exploitation scenario constructed for each one. The scenario was then used as a basis for calculating the maximum rate at which the channel could be employed. The results of this analysis are contained in the report [20].

7.2.2 Capacity Limitation Techniques

STOP 5.2.E employs two covert channel capacity limitation techniques: time delay and the introduction of noise.

7.2.2.1 Resource Exhaustion Delay

The resource exhaustion delay is used to reduce covert storage channel capacity without eliminating any user capabilities. The delay is imposed only when a resource exhaustion error occurs. The resource exhaustion

delay is a site-set time interval (the default is 102.4 seconds) that puts processes to sleep for a set amount of time when a resource exhaustion condition is encountered. The param_edit command is used to change the default value. Processes that use branch blocks, data blocks, memory tables, or large page tables trigger the delay when system resources are exhausted. The use of the delay generates an auditable event.

7.2.2.2 Introduction of Covert Channel Noise

The ability of one process to determine the behavior of another based on consecutive values of unique identifiers (uid) is a known channel in trusted systems. On XTS-300, this channel is controlled by randomizing uids, effectively introducing noise into the process.

Two uid generation algorithms are used to name the fundamental TCB objects – devices, processes, semaphores and segments. One algorithm is used for devices, segments and semaphores, and the other for processes. Since devices may be created and removed only by trusted software, a potential channel exists only for segments, semaphores and processes.

The randomization of process uids is uniformly distributed across the entire range of such uids. For segment uids, however, Wang chose a technique that limits the rate at which the channel operates. Semaphore uids use the same randomization technique as segment uids but an auditable delay is added similar to the resource exhaustion delay in order to limit the bandwidth below ten bps.

7.3 Design Specification and Verification

The vendor used the Bell and LaPadula model [4] with the Biba model [5] as the abstract model and its Multics Interpretation [4] as the concrete model. The Biba model was incorporated into the Bell and LaPadula model through a modification of the definition of the dominates relation.

The system allows four access modes: read, write, execute, and search. The read and execute modes map to the read mode in the Bell and LaPadula model. The search mode is a restricted form of read mode in the Bell and LaPadula model. It only applies to the discretionary access control decisions; for the mandatory access control, read access is required. The write mode maps to the append mode in the Bell and LaPadula model.

- The system state has four components: current access set, access permission matrix, level functions, and object hierarchy.
- The current access set consists of the tuples (subject, object, access mode).
- The access permission matrix contains access modes allowed to each subject for each object.
- Three level functions provide subject maximum clearance level, current subject MAC label, and current object MAC label.
- The object hierarchy ensures that the MAC label of an object dominates the MAC label of the object's parent.

The model has three axioms (policies). The first policy is called the simple security policy and it ensures that a subject can only read the objects whose MAC label is dominated by the maximum clearance of the subject. The second policy is called the security star property and it ensures that a subject can read an object only if the subject's current MAC label dominates the object's MAC label. This policy further ensures that a subject can only write to an object if the MAC label of the object dominates the current MAC label of the subject. The third policy is the discretionary security policy, which ensures that a subject can only access an object in the access modes explicitly granted to the subject for that object.

The model uses 11 rules of operation. Of the rules, five manipulate the current access set; they are: get read access, get append access, get execute access, get write access, and release read/append/execute/write access. Two rules manipulate the access permission matrix: give read/append/execute/write access, and rescind read/append/execute/write access. Two manipulate the level functions. These rules are: change subject current MAC label, and change object MAC label. Two others manipulate the object hierarchy: create object, and delete object.

The vendor has identified 34 state-transforming TCB interface operations. For each of these 34 operations, the vendor provided the following in the English language: interface description, security checks from the MAC and DAC viewpoints, state-transformation rules, and restrictions in terms of constraints in addition to the MAC and DAC checks (e.g., required capabilities to successfully invoke the operation). The rules are translated into a sequence of the rules based on the 11 Multics Interpretation rules. Some of the rules are described as the variants to the original 11 rules for one of the following reasons: some objects (e.g., temporary segments) are not members of the object (file) hierarchy, or changes to the access control matrix are made without altering current accesses. Several TCB interfaces are not modeled because they are state-preserving for one of the following reasons: obtaining object attributes, setting unmodeled (not security-relevant) attributes of objects (MAC and DAC policies are still enforced on the objects), internal TCB implementation functions, and object-referencing (I/O) functions.

The vendor has provided a Descriptive Top-Level Specification (DTLS) [16] that describes the TCB interface operations through the kernel gates, the TSS gates, and the trusted commands. In combination with the Trusted Programmer's Reference Manual [34] and User's Manual [35], each interface description includes the following: function of the interface, inputs, outputs, security checks made by the TCB, and the errors and exceptions reported by the TCB. The DTLS is complete and consistent with respect to the security model.

7.4 TCB Recovery

The system recovery after a failure is accomplished by rebooting. After a failure, the filesystems may not be in a secure state, because the segments in memory and on the disk may not have been synchronized at the time of the system failure. To ensure that a filesystem is in a secure state, the system provides two trusted commands: check and fscheck. For a detailed description of these commands, see Section 4.7.3.2.2, page 69. When a filesystem is mounted, the kernel sets the "mounted" flag in the super page of the filesystem. The kernel resets this flag upon unmounting a filesystem. The kernel does not mount a filesystem if the "mounted" flag is already set.

The super page of a filesystem also contains a flag called "checked." This flag is set by the check program after performing segment level repair. The fscheck program will not run unless the "checked" flag is set, i.e., the check program has been run. The fscheck program performs the hierarchical file system level repair and resets the "checked" and "mounted" flags. Thus, all mounted filesystems must have the check and the

fscheck programs run against them after a system failure. The check program must be run before the fscheck program. The check and the fscheck programs ensure that the segments and filesystems are in a secure state.

7.5 Configuration Management

The vendor has produced manuals [30, 19] that describe the configuration management system employed and the processes by which the system is maintained during each Rating Maintenance Phase (RAMP), respectively.

Because the base hardware used for the XTS-300 is comprised of COTS products, a method of keeping track of revisions is mandatory. Some third-party vendors provide change notification in advance. Others provide little or no notice. Wang keeps a list of accepted components by serial, model, and revision number. As each component is received, it is inspected visually to determine whether the component numbers are the same as those on the list. When a change is detected, Wang requests change information from the third-party vendor. Regardless of whether the change information is received from the third-party vendor, a subset of the test suite is always run on the assembly room floor. If all tests pass, the new component revision is added to the acceptable product list.

The Hardware Engineering Group (HWEG) is responsible for evaluating and testing any hardware changes prior to acceptance into the XTS-300 product. The HWEG performs tests (in addition to those performed by Intel) to ensure that a standard product change does not have an effect on XTS-300. If a change is determined to have an impact on XTS-300, additional testing using STOP 5.2.E is performed, the extent of which is determined by the Software Configuration Review Board (CRB).

The software configuration management for STOP 5.2.E follows a formal change reporting and review procedure. When a discrepancy is reported or an enhancement required, it is usually entered into the Problem Report (PR) database. When work is ready to commence on an existing PR or an immediate problem or enhancement is needed, an analyst is assigned (usually one of the senior members of a functional software development area) who prepares a report in the Internal Software Note (ISN) database summarizing the situation and the proposed action.

The report is evaluated and is reviewed by management. A copy of the ISN is then circulated to members of the Software Configuration Control Board (CCB). This board, which communicates by email, consists of lead analysts from the software development team, as well as representatives from HWEG, and engineering management.

It is the CCB's responsibility to evaluate the impact of the proposed change and to verify that the proposed changes are comprehensive and technically sound. This review of the CCB is viewed as preliminary and advisory in nature. If a developer starts work before CCB approval, or ignores the advice of the CCB, then they risk having spent time on wasted effort when the next phase, the Change Review Board (CRB), reviews the work actually performed by the developer.

After development work is completed, the more important phase of the review process begins. This second review is performed by the CRB and consists of a review of the modified code and all relevant documentation, testing, and performance analysis ensuring that they are adequately covered[1]. Standard software tools (e.g., Source Code Control System (SCCS), lint, diff) are utilized in order to organize the changes and make them more convenient for the CCB to analyze.

[1] The full procedure involving the CRB is utilized, even for "emergency" fixes.

The CRB makes the final decision with respect to all changes to STOP 5.2.E.

All source code, test code and design documentation is maintained under SCCS.

7.6 System Integrity

XTS-300 has a multitude of tests that can be executed to test the integrity of the hardware. The Wang provides three categories of tests: COTS offline tests, BIOS power-on self-tests (POST), and Wang-written tests.

The COTS off-line tests include, but are not limited to, the testing of many CPU instructions, known bugs in earlier CPU chips, FPU instructions, repeated reads and writes of different patterns to all memory, checks for bad address lines, all DMA channels and registers, interrupt controller priorities and vectoring, parallel controller reads and writes, ethernet controller, disk, tape and CD-ROM tests, and invalid opcode tests.

The BIOS POST test is performed prior to loading STOP 5.2.E. The POST performs both security relevant and non-security relevant tests. Of the security relevant tests, the following tests are included: reads and writes of several values into all CPU registers, reads and writes to DMA registers, reads and writes of data to all motherboard and adapter card memory locations, reads and writes to all main memory, reads and writes of data to the CMOS RAM, data cache and cache disable, the keyboard interface, Programmable Interrupt Timer (PIT) channels 0, 1, and 2, simple display features, and entering and exiting protected mode.

In addition to BIOS POST, some hardware controllers perform self-tests in response to the hardware "reset" condition which propagates through the system during every reboot. These tests are performed by the SCSI host adapter, the SCSI tape drive, the SCSI CD-ROM drive, and the SCSI disk drive.

The Wang-written tests are a subset of the existing security test suite for the XTS-300. These tests are run off-line from a special bootable STOP 5.2.E diskette which is provided to all XTS-300 customers. The tests cover the following areas: segmentation, I/O protection, privileged instructions, control transfers, trap/interrupt/fault handling, and tasking.

7.7 Testing

Wang has provided the security analysis team with a test plan [26] describing their approach to testing the system and detailing the test procedures [27, 28] developed to exercise all TCB interfaces over as full a range of parameter values and boundary conditions as was practical[2]. The security analysis team reviewed the vendor's Test Procedures documents that describe each of the individual tests developed by the vendor. Additionally, the team conducted their own set of penetration tests, the conclusions of which are summarized in Section 8.20, page 128.

The XTS-300 system is continuously exercised outside of the formal testing environment through activities described below. Some application development (primarily trusted) is being produced and maintained on production XTS-300 systems at Wang. As further evidence that the system is usable and functional outside of the formal testing environment, the XTS-300 system is currently operational at several sites including the

[2] The vendor also relied on code path analysis that was performed during the original evaluation to provide assurance that data was passed correctly from TSS to the kernel [15, 14].

Federal Bureau of Investigations (FBI), the Canadian Government, the National Security Agency (NSA), National Research Laboratory (NRL), Pentagon (Joint Staff J6), Sandia National Laboratories, Naval Post Graduate School, DoD's Defense Message System (DMS), NIMA, US CENTCOM, various sites in the Intelligence Community, United States Space Command (USSPACECOM) and North American Defense Command (NORAD), Wright Patterson Air Force Base(WPAFB) and the United States Forces in Korea (USFK) primarily in network guard applications.

7.8 Architecture Study

A formal architecture study was conducted during the original evaluation and documented in [14]. The security analysis team relied on the results of the original study and the study conducted for STOP 4.1 and STOP 4.4.2 to determine the extent to which the architecture of the XTS-300 running STOP 5.2.E was examined. The security analysis team conducted a detailed study of the code that changed from STOP 4.4.2 to STOP 5.2.E, which represents approximately ten percent of the TCB. The team examined the code primarily to be sure that the XTS-300 still satisfied the B3 requirements for system architecture. The team also analyzed the design documentation to confirm the correspondence between the implementation and the design documentation.

A formal architecture study was performed during the last week of October. The security analysis team did the following for the code that was inspected:

- Assess the degree and manner of interaction of the module with other modules, to judge to what extent it supported the aim of a modular TCB.
- Assess the degree and manner of interaction of each function within the module with other functions.
- Analyze the usage of global variables by the module, and the functions within the module.
- Examine how well the goal of data hiding was achieved.
- Consider the complexity and comprehensibility of the module, and the functions within the module.
- Investigate whether the use of any module or function depended on side-effects of that module or function.
- Check the accuracy and the degree of correspondence of the design documentation and source code.
- Check the code for conformance with Wang's coding standards [18].
- Search for duplicate code.
- Search for duplicate data.
- Consider whether privileges were acquired only when needed by modules, and relinquished when no longer needed.

The security analysis team concluded that the B3 requirements for system architecture are satisfied.

As was the case during the original evaluation, the team used the detailed design documentation, rather than the DTLS, as the basis for establishing correspondence with the implementation. The detailed design

specification were used because they contained detail absent from the DTLS and because Wang regards them as the primary documentation of the system design.

Chapter 8

Evaluation as a B3 System

8.1 Discretionary Access Control

Requirement

The TCB shall define and control access between named users and named objects (e.g., files and programs) in the ADP system. The enforcement mechanism (e.g., access control lists) shall allow users to specify and control sharing of those objects, and shall provide controls to limit propagation of access rights. The discretionary access control mechanism shall, either by explicit user action or by default, provide that objects are protected from unauthorized access. These access controls shall be capable of specifying, for each named object, a list of named individuals and a list of groups of named individuals with their respective modes of access to that object. Furthermore, for each such named object, it shall be possible to specify a list of named individuals and a list of groups of named individuals for which no access to the object is to be given. Access permission to an object by users not already possessing access permission shall only be assigned by authorized users.

Applicable Features

The XTS-300 Trusted Computer Base (TCB) allows named users to define and control access to named objects through the use of an Access Control List (ACL). Every subject in XTS-300 has associated with it an effective user and group; every named object has an ACL.[1] Each ACL contains permissions that specify the allowable access for the owning user, the owning group, up to six other users or groups, and any user or group not explicitly listed. These permissions can either grant or deny a particular form of access to a named object. When a subject introduces an object into its address space, the ACL is checked to ensure that the subject can access the object. If the ACL is changed, XTS-300 forces revalidation of access.

When an object is created, the default discretionary access control (DAC) is either specified by the creator or restricted to the creator. File system objects have an additional level of default protection in that a subject must be able to access the directory to access the objects contained therein. Propagation of access rights is controlled by restricting the ability to change the owning user or group of an object to the object's owner or a user with the appropriate privilege.

For more information on the discretionary access control mechanisms provided in XTS-300, see Section 6.2.2, page 87.

[1] Sockets do not have an explicit ACL but an implicit ACL of the shared memory segment associated with them. This ACL only gives R/W access to the owner or an attached process.

Conclusion

XTS-300 satisfies the B3 Discretionary Access Control requirement.

8.2 Object Reuse

Requirement

All authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation or reallocation to a subject from the TCB's pool of unused storage objects. No information, including encrypted representations of information, produced by a prior subject's actions is to be available to any subject that obtains access to an object that has been released back to the system.

Applicable Features

Segments, TCB internal data structures, registers, device controllers, and directory entries are handled correctly in terms of object reuse. Tapes are properly handled through administrative techniques and degaussing. A more complete description of object reuse under STOP 5.2.E is discussed on Section 6.7, page 99.

Conclusion

XTS-300 satisfies the B3 Object Reuse requirement.

8.3 Labels

Requirement

Sensitivity labels associated with each ADP system resource (e.g., subject, storage object, ROM) that is directly or indirectly accessible by subjects external to the TCB shall be maintained by the TCB. These labels shall be used as the basis for mandatory access control decisions. In order to import non-labeled data, the TCB shall request and receive from an authorized user the security level of the data, and all such actions shall be auditable by the TCB.

Applicable Features

Every identified storage object in XTS-300 has associated with it a Mandatory Access Control (MAC) label (consisting of a sensitivity label and an integrity label) that is maintained by the TCB. These labels are used by the TCB to enforce the requirements of the XTS-300 mandatory access policy. Non-labeled data can only be imported in XTS-300 via a single-level logical device; the TCB uses the MAC label assigned to

the device to label the imported data. The action of a user obtaining initial access to a device is audited, as is the action of an administrator establishing the current MAC label of the device.

For more information, see Section 5, page 83, for a discussion of the labels associated with objects in XTS-300; Section 6.6, page 95, describes auditing in XTS-300.

Conclusion

XTS-300 satisfies the B3 Labels requirement.

8.4 Label Integrity

Requirement

Sensitivity labels shall accurately represent security levels of the specific subjects or objects with which they are associated. When exported by the TCB, sensitivity labels shall accurately and unambiguously represent the internal labels and shall be associated with the information being exported.

Applicable Features

All MAC labels (which consist of both sensitivity and integrity labels) in XTS-300 are maintained by the TCB, which protects the labels from unauthorized modification. These labels are assigned initially based on the current MAC label of the creating subject, which is in turn derived from the default MAC label of the user on whose behalf the subject is operating. This default MAC label, which is dominated by the user's clearance, is initially assigned to the user by the administrator via ua_edit; a user with the appropriate capability may change this default level via the change default level (cdl) command.

The only multilevel device to which labels are exported in XTS-300 are disk devices being accessed as filesystems. Filesystems are maintained by the XTS-300 TCB, and use the same segment structure as is possessed by the segment when in memory. The sensitivity label for a segment written to a file system will also be written to that filesystem (as part of the Segment Branch Table Entry (SBTE)). There is one exception to this rule: temporary and shared memory segments may be written to the boot filesystem while their SBTEs are held in memory. However, temporary segments are deleted once they are no longer mapped into any process' address space, while shared memory segments are not preserved across bootloads.

In terms of single-level devices, labels are also exported to tapes by the trusted fsave program. The tape itself is labeled at the MAC label of the fsave program, which must dominate the upper end of the MAC label range of the filesystem being saved. The fsave program writes tapes using the segment block structure of the filesystem; labels are written in their internal format as branch blocks.

For more information, see Section 6.2.1, page 86.

Conclusion

XTS-300 satisfies the B3 Label Integrity requirement.

8.5 Exportation of Labeled Information

Requirement

The TCB shall designate each communication channel and I/O device as either single-level or multilevel. Any change in this designation shall be done manually and shall be auditable by the TCB. The TCB shall maintain and be able to audit any change in the current security level or levels associated with a communication channel or I/O device.

Applicable Features

In the XTS-300 TCB, all non-filesystem I/O devices are single-level (filesystem devices are single-level only when they are accessed as a device and not as a filesystem). Filesystems can either be multilevel or single-level. Filesystems possess ranges and are single-level if the minimum and maximum MAC labels are equal. Trusted programs are provided to allow users with appropriate privilege to change the designated ranges of filesystems (config_edit) and the designated level of a single-level device (sda). Filesystems, although multi-level, cannot have this range changed; is it specified at the time the filesystem is created by an administrator via mkfsys. All possible changes are audited.

More information can be found in the following sections:

- Section 4.7.3.2, page 68, provides a discussion of the trusted commands used to manipulate device-related labels.
- Section 6.6, page 95, provides a discussion of the auditing associated with device label changes.

Conclusion

XTS-300 satisfies the B3 Exportation of Labeled Information requirement.

8.6 Exportation to Multilevel Devices

Requirement

When the TCB exports an object to a multilevel I/O device, the sensitivity label associated with that object shall also be exported and shall reside on the same physical medium as the exported information and shall be in the same form (i.e., machine-readable or human-readable form). When the TCB exports or imports

an object over a multilevel communication channel, the protocol used on that channel shall provide for the unambiguous pairing between the sensitivity labels and the associated information that is sent or received.

Applicable Features

The only multilevel device supported by XTS-300 are disk devices accessed as filesystems. Filesystems use the same segment structure as is used in memory. Access control information (including the MAC label) is maintained in the SBTEs, which are stored on the same filesystem as the segments they describe. When a segment is brought into memory by the kernel File System Manager, these MAC labels are an integral part of the information transmitted.

When a disk is accessed as a raw device, as is done when a filesystem is created and checked, it is treated as a single-level device. In these operations, the program accessing the device is trusted to maintain label integrity. fsave and frestore are trusted to properly maintain the labels associated with the segments written to tape.

For more information, see Section 4.5.5.2, page 51.

Conclusion

XTS-300 satisfies the B3 Exportation to Multilevel Devices requirement.

8.7 Exportation to Single-Level Devices

Requirement

Single-level I/O devices and single-level communication channels are not required to maintain the sensitivity labels of the information they process. However, the TCB shall include a mechanism by which the TCB and an authorized user can reliably communicate to designate the single security level of information imported or exported via single-level communication channels or I/O devices.

Applicable Features

In order to communicate the MAC label of a non-filesystem device (all non-filesystem devices are treated by XTS-300 as single-level), the XTS-300 TCB provides the administrator, through the trusted path, with the set device access (sda) command. This command is used to designate the single MAC label (which includes the sensitivity label) of the data imported from or exported to the logical device. Any changes to the MAC label of a device are audited.

For more information, see Section 4.7.3.2.10, page 72.

Conclusion

XTS-300 satisfies the B3 Exportation to Single-Level Devices requirement.

8.8 Labeling Human-Readable Output

Requirement

The ADP system administrator shall be able to specify the printable label names associated with exported sensitivity labels. The TCB shall mark the beginning and end of all human-readable, paged, hardcopy output (e.g., line printer output) with human-readable sensitivity labels that properly [2] represent the sensitivity of the output. The TCB shall, by default, mark the top and bottom of each page of human-readable, paged, hardcopy output (e.g., line printer output) with human-readable sensitivity labels that properly represent the overall sensitivity of the output or that properly represent the sensitivity of the information on the page. The TCB shall, by default and in an appropriate manner, mark other forms of human-readable output (e.g., maps, graphics) with human-readable sensitivity labels that properly represent the sensitivity of the output. Any override of these marking defaults shall be auditable by the TCB.

Applicable Features

The XTS-300 TCB labels all printed output with banner pages (at the beginning and end of each file) and internal page labels (at the top and bottom of each page) that reflect the sensitivity label associated with the file being printed. These banner pages cannot be suppressed. A user with the UNMARKED PRINT ALLOWED capability can suppress the printing of internal page labels (which results in a distinct auditable event), but cannot change the internal page labels.[3] The human-readable forms of the levels and categories that make up the sensitivity label are obtained from the Security Map database, which is maintained by the administrator using the sm_edit command. The length of the printer page cannot be shortened; the standard page length does allow space for the maximum sensitivity label.

Spoofing is prevented through a combination of system and procedural controls. The header and trailer banners contain a sequence number which is not alterable by a user. The sequence number is represented internally by a 16-bit unsigned integer. Operators and administrators are advised by the Trusted Facility Manual [33] to inspect the printed output to be sure that the sequence numbers are in proper order. If a user tries to create a false banner page, then the sequence numbering would be out of order. These header and trailer banner pages will always be generated and cannot be manipulated by a user.

The evaluation configuration for XTS-300 does not support the ability to produce other forms (e.g., maps, graphics) of human-readable output.

For more information, see Section 4.7.1.6, page 63.

[2] The hierarchical classification component in human-readable sensitivity labels shall be equal to the greatest hierarchical classification of any of the information in the output that the labels refer to; the non-hierarchical category component shall include all of the non-hierarchical categories of the information in the output the labels refer to, but no other non-hierarchical categories.

[3] Supressing page markings is the only way for an application to send printer specific escape sequences to the printer. Otherwise, such sequences are filtered.

Conclusion

XTS-300 satisfies the B3 Labeling Human-Readable Output requirement.

8.9 Subject Sensitivity Levels

Requirement

The TCB shall immediately notify a terminal user of each change in the security level associated with that user during an interactive session. A terminal user shall be able to query the TCB as desired for a display of the subject's complete sensitivity label.

Applicable Features

XTS-300 does not change the MAC label of a subject automatically. A user must explicitly request the TCB to change the current MAC label maintained for the user by the Secure Server that is used for Secure Server operations and for labeling subsequent process creation. The ability to change this label is restricted by the capability mechanism. At any time, the user can issue the sl command to the TCB to ascertain the current MAC label of the Secure Server. Note that, once created, a process cannot change its MAC label.

Conclusion

XTS-300 satisfies the B3 Subject Sensitivity Levels requirement.

8.10 Device Labels

Requirement

The TCB shall support the assignment of minimum and maximum security levels to all attached physical devices. These security levels shall be used by the TCB to enforce constraints imposed by the physical environments in which the devices are located.

Applicable Features

Every physical device in XTS-300 communicates with the TCB through a controller for the appropriate device type. The TCB identifies devices through the use of major and minor numbers. The major number identifies the device type and is associated with a controller. The minor number identifies the particular device on the controller (logical device, if the device supports a filesystem and is partitioned). The TCB maintains the major and minor number information in the LDD. This information includes the minimum and maximum MAC labels that may be possessed by information flowing to and from the device. The TCB

enforces the restriction that any logical devices accessed via a major and minor number have a MAC label that is within the defined range for the device.

Conclusion

XTS-300 satisfies the B3 Device Labels requirement.

8.11 Mandatory Access Control

Requirement

The TCB shall enforce a mandatory access control policy over all resources (i.e., subjects, storage objects, and I/O devices) that are directly or indirectly accessible by subjects external to the TCB. These subjects and objects shall be assigned sensitivity labels that are a combination of hierarchical classification levels and non-hierarchical categories, and the labels shall be used as the basis for mandatory access control decisions. The TCB shall be able to support two or more such security levels. The following requirements shall hold for all accesses between all subjects external to the TCB and all objects directly or indirectly accessible by these subjects: A subject can read an object only if the hierarchical classification in the subject's security level is greater than or equal to the hierarchical classification in the object's security level and the non-hierarchical categories in the subject's security level include all the non-hierarchical categories in the object's security level. A subject can write an object only if the hierarchical classification in the subject's security level is less than or equal to the hierarchical classification in the object's security level and all the non-hierarchical categories in the subject's security level are included in the non-hierarchical categories in the object's security level. Identification and authentication data shall be used by the TCB to authenticate the user's identity and to ensure that the security level and authorization of subjects external to the TCB that may be created to act on behalf of the individual user are dominated by the clearance and authorization of that user.

Applicable Features

The XTS-300 TCB enforces a mandatory access control policy over all identified system resources (i.e., subjects, storage objects, and I/O devices) that are accessible, either directly or indirectly, to subjects external to the TCB. This policy is a combination of the Bell and LaPadula [4] and strict Biba [5] models. It uses, as the basis of its enforcement, MAC labels that are associated with every subject and object in the system. These MAC labels consist of hierarchical sensitivity and integrity levels (16 sensitivity, 8 integrity), and nonhierarchical sensitivity and integrity categories (64 sensitivity, 16 integrity).

XTS-300 provides a *dominates* function that is used to compare sensitivity or integrity labels; this comparison is done whenever a subject external to the TCB accesses an object. To read an object, the sensitivity label of the subject must dominate the sensitivity label of the object, and the integrity label of the object must dominate the integrity label of the subject. In order to write an object, the sensitivity label of the object must dominate the sensitivity label of the subject, and the integrity label of the subject must dominate the integrity label of the object. This is illustrated in Figure 6.1 (page 85).

Every user in XTS-300 has an identification and authentication database record that specifies the MAC label

of the user's clearance. The TCB enforces the restriction that any subject created on behalf of a user has a current MAC label dominated by the user's clearance.

For more information, see Section 6.2.1, page 86.

Conclusion

XTS-300 satisfies the B3 Mandatory Access Control requirement.

8.12 Identification and Authentication

Requirement

The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. Furthermore, the TCB shall maintain authentication data that includes information for verifying the identity of individual users (e.g., passwords) as well as information for determining the clearance and authorizations of individual users. This data shall be used by the TCB to authenticate the user's identity and to ensure that the security level and authorizations of subjects external to the TCB that may be created to act on behalf of the individual user are dominated by the clearance and authorization of that user. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual ADP system user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.

Applicable Features

STOP 5.2.E requires all users to identify and authenticate themselves before they are allowed to access system resources. Users enter unique usernames and passwords to identify and authenticate themselves to the system.

The TCB maintains authentication data including username, password, and the default and maximum security and integrity levels for each user in the User Access Authentication database. The only user that this database is accessible to is the system administrator. This database is at maximum security and maximum integrity. Additionally, this file is protected from access by untrusted software by assigning it a unique file subtype.

This data is used by the Secure Server to authenticate a user's identity and to verify that the user's default security and integrity levels are within the maximum security and integrity levels allowed for that user, as well as being within the range of allowed levels for the given terminal.

Each individual is associated with a unique identifier associated with that individual for recording all auditable actions taken by that user.

Conclusion

XTS-300 satisfies the B3 Identification and Authentication requirement.

8.13 Trusted Path

Requirement

The TCB shall support a trusted communication path between itself and users for use when a positive TCB-to-user connection is required (e.g., login, change subject security level). Communications via this trusted path shall be activated exclusively by a user or the TCB and shall be logically isolated and unmistakably distinguishable from other paths.

Applicable Features

The TCB establishes a trusted path with a user when the Secure Attention Key (SAK) on the terminal is pressed (the <BREAK> key). The TCB displays the login banner if no user is currently logged in at that terminal. If a user is logged in at that terminal, the TCB displays the current process family identifier and MAC label information. The path is logically isolated since the establishment of the trusted path by processing the SAK is initiated by the terminal driver as soon as the SAK is detected. The terminal is then under the control of the TCB; from then on, no untrusted processes can perform I/O to the terminal. The path is unmistakable since it is only initiated by the user action of pressing the SAK, and the TCB is in control of the terminal after the SAK is detected by the terminal driver.[4] The TCB never initiates a trusted path without the user pressing the SAK. Once the trusted path is established by the TCB, the user can enter any of the Trusted Commands in Section 4.7.3, page 64.[5]

Conclusion

XTS-300 satisfies the B3 Trusted Path requirement.

8.14 Audit

Requirement

The TCB shall be able to record the following types of events: use of identification and authentication mechanisms, introduction of objects into a user's address space (e.g., file open, program initiation), deletion of objects, and actions taken by computer operators and system administrators and/or system security officers. The TCB shall also be able to audit any override of human-readable output markings. For each

[4] Note that trusted path cannot be initiated from a serial mouse or across a network.

[5] Since the XTS-300 gives untrusted applications two seconds to clean up after SAK is depressed, users should wait at least that long to ensure that they are really communicating with the TCB.

recorded event, the audit record shall identify: date and time of the event, user, type of event, and success or failure of the event. For identification/authentication events the origin of request (e.g., terminal ID) shall be included in the audit record. For events that introduce an object into a user's address space and for object deletion events the audit record shall include the name of the object and the object's security level. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity and/or object security level. The TCB shall be able to audit the identified events that may be used in the exploitation of covert storage channels. The TCB shall contain a mechanism that is able to monitor the occurrence or accumulation of security auditable events that may indicate an imminent violation of security policy. This mechanism shall be able to immediately notify the security administrator when thresholds are exceeded, and if the occurrence or accumulation of these security relevant events continues, the system shall take the least disruptive action to terminate the event.

Applicable Features

The Audit facility within STOP 5.2.E is used by the TCB to record security-relevant events that take place on the system.

The File System Daemon process produces audit files in the `/audit` directory from information it obtains from the kernel. The audit files are protected with file subtypes that prevent access by untrusted software.

Audit events are generated by Trusted Software, TCB System Services, and the kernel and include the following types of events:

- Login attempted
- Logout command issued
- Opens and closes of file system objects
- Creates and deletes of file system objects
- Operator command issued
- Administrator command issued
- Print request issued with no markings

Each audit record has a header that contains the size of the audit record, type of event being audited, date and time the audit record was generated, process ID of the process causing the audit event, MAC label of the process, effective privileges of the process, real user ID, and real group ID.

Each audit record also contains data pertinent to the particular audit event including , but not limited to, the device id, MAC labels (including both the new and old labels in the case of a change), file system ID, segment name, channel number, and privilege set.

Audit functions can be selectively audited by event, by user and by object security level. The auditing mechanism is implemented such that a minimum MAC label may be specified, below which audit messages are not generated for object creation, deletion, and access.

STOP 5.2.E monitors as imminent security violations the accumulation of repeated login failures. The number of failures allowed before action is taken is site configurable. After this number of failed login

attempts, a message is sent to the system console and the terminal is locked out. That terminal is no longer usable until either that terminal is freed up via the ctl command or the duration of timeout has expired.

Conclusion

XTS-300 satisfies the B3 Audit requirement.

8.15 System Architecture

Requirement

The TCB shall maintain a domain for its own execution that protects it from external interference or tampering (e.g., by modification of its code or data structures). The TCB shall maintain process isolation through the provision of distinct address spaces under its control. The TCB shall be internally structured into well-defined largely independent modules. It shall make effective use of available hardware to separate those elements that are protection-critical from those that are not. The TCB modules shall be designed such that the principle of least privilege is enforced. Features in hardware, such as segmentation, shall be used to support logically distinct storage objects with separate attributes (namely: readable, writeable). The user interface to the TCB shall be completely defined and all elements of the TCB identified. The TCB shall be designed and structured to use a complete, conceptually simple protection mechanism with precisely defined semantics. This mechanism shall play a central role in enforcing the internal structuring of the TCB and the system. The TCB shall incorporate significant use of layering, abstraction and data hiding. Significant system engineering shall be directed toward minimizing the complexity of the TCB and excluding from the TCB modules that are not protection-critical.

Applicable Features

XTS-300 employs a ring architecture that uses hardware to enforce domain isolation. From the innermost ring, Ring 0, to the outermost, Ring 3, the ability of the software to access system objects decreases. The TCB elements are clearly defined as the Security Kernel, TCB System Services (TSS), and Trusted Software. Each element of the TCB executes in a separate domain. Ring 0 comprises the Kernel domain, Ring 1 comprises the Trusted System Services domain, and, for a TCB process, Ring 2 comprises the Operating Systems Services domain, which includes that software requiring privilege and/or integrity to execute. For untrusted software, Commodity Application System Services (CASS) runs in Ring 2. User application software runs in Ring 3, the domain of least access. The ring architecture is also used to enforce write, read, and call (execute) access.

Each process has a unique, virtual address space using hardware segments. Only the text for the kernel, TSS, and CASS plus kernel data, such as the memory map, are considered global information. Each process has its own copy of ring stacks, process descriptor segments, and process-specific text and data. Least privilege is enforced by tailoring privileges and integrity to particular security-relevant operations and to the software which needs to perform such operations. The kernel performs as a reference monitor and as a basic operating system. The kernel manipulates those entities which are objects (e.g., segments) and its internal data structures. TSS provides the file system hierarchy and performs high-level user I/O functions. Trusted

Software provides the user interface to perform security-relevant functions and the trusted path. Wang has published [16] and [25] which define the components of the TCB and its interface. The TCB enforces a strict interpretation of the Bell and LaPadula security model [4] and the Biba integrity model [5].

Segmentation is used in STOP 5.2.E to support logically distinct storage objects. The permissions set for each global segment are appropriate, given the contents. TCB text is protected from reading and writing. Data internal to a domain is manipulated only by the code running in that domain. In addition, the modules within a domain have been designed to manipulate only those data structures pertinent to the functions within a particular module. The kernel and TSS are internally layered such that, in general, those routines that rely on others to perform services are at higher layers. Routines which do not rely on others to perform functions are placed at lower layers of the kernel and TSS, respectively. In addition, the TCB contains only that code necessary to perform its functionality. The TCB of XTS-300 is modular and relatively easy to understand.

Conclusion

XTS-300 satisfies the B3 System Architecture requirement.

8.16 System Integrity

Requirement

Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB.

Applicable Features

XTS-300 has a multitude of tests that can be executed to test the integrity of the hardware. Wang provides three categories of tests: COTS offline tests, BIOS power-on self-tests (POST), and Wang-written tests.

For a more detailed description of system integrity features, see Section 7.6, page 110.

Conclusion

XTS-300 satisfies the B3 System Integrity requirement.

8.17 Covert Channel Analysis

Requirement

The system developer shall conduct a thorough search for covert channels and make a determination (either by actual measurement or by engineering estimation) of the maximum bandwidth of each identified channel.

Applicable Features

The vendor conducted a thorough analysis of the system and produced a covert channel analysis [20] (see Section 7.2, page 105). In it, the vendor describes the technique used to ensure the search was thorough, and describes exploitation scenarios for each of the channels uncovered. Using these exploitation scenarios, a calculation is made as to the maximum capacity of each of the channels.

Conclusion

XTS-300 satisfies the B3 Covert Channel Analysis requirement.

8.18 Trusted Facility Management

Requirement

The TCB shall support separate operator and administrator functions. The functions performed in the role of a security administrator shall be identified. The ADP system administrative personnel shall only be able to perform security administrator functions after taking a distinct auditable action to assume the security administrator role on the ADP system. Non-security functions that can be performed in the security administration role shall be limited strictly to those essential to performing the security role effectively.

Applicable Features

The system supports the operator and administrator functions through the use of integrity labels. Any user with an integrity label at or higher than the operator integrity level can execute the operator commands, provided the user has the required capabilities.

Any user with an integrity label at or higher than the administrator integrity level can perform the administrator commands, provided the user has the required capabilities. The administrator integrity level is higher than the operator integrity level. Thus, a user with the operator integrity level cannot execute the administrator commands.

The Trusted Facility Manual (TFM) recommends that the untrusted users' assigned maximum integrity levels should be lower than the Operating System Services (OSS) integrity level (the OSS integrity level

is lower than the operator integrity level). Thus, an untrusted user cannot execute the operator or the administrator commands.

The two ways for a user to assume the role of the administrator is to login with administrator or higher integrity, or to use the sl command to set the integrity label to administrator or higher. Since all user logins and the use of the sl command are auditable along with the MAC label, the assumption of the administrator role is auditable.

A user logged in with the OSS or higher integrity level cannot execute the run command (which is used for running untrusted programs). This feature limits the user acting in the administrator role (since the user's integrity level is higher than OSS integrity level) to trusted commands only.

Conclusion

XTS-300 satisfies the B3 Trusted Facility Management requirement.

8.19 Trusted Recovery

Requirement

Procedures and/or mechanisms shall be provided to assure that, after an ADP system failure or other discontinuity, recovery without a protection compromise is obtained.

Applicable Features

After a system failure, the filesystems may not be in a consistent and secure state. The system provides two trusted commands to bring a filesystem to a secure state. The check command repairs the segment level file system and the fscheck command repairs the hierarchical file system (see Section 4.7.3.2.2, page 69). After a system failure, a filesystem cannot be referenced, used, or remounted until both the check and the fscheck commands are executed.

Conclusion

XTS-300 satisfies the B3 Trusted Recovery requirement.

8.20 Security Testing

Requirement

The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. A team of individuals who thoroughly understand the specific implementation of the TCB

shall subject its design documentation, source code, and object code to thorough analysis and testing. Their objectives shall be: to uncover all design and implementation flaws that would permit a subject external to the TCB to read, change, or delete data normally denied under the mandatory or discretionary security policy enforced by the TCB; as well as to assure that no subject (without authorization to do so) is able to cause the TCB to enter a state such that it is unable to respond to communications initiated by other users. The TCB shall be found resistant to penetration. All discovered flaws shall be corrected and the TCB retested to demonstrate that they have been eliminated and that new flaws have not been introduced. Testing shall demonstrate that the TCB implementation is consistent with the descriptive top-level specification. No design flaws and no more than a few correctable implementation flaws may be found during testing and there shall be reasonable confidence that few remain.

Applicable Features

As a basis for comparison, the discussion of the security testing requirement from the evaluation of STOP 3.1.E was retained. The results of the evaluation of STOP 5.2.E are described in the paragraph following the one below.

The security mechanisms of STOP 5.2.E were subjected to both functional and penetration testing and found to work as claimed in the system documentation. The team conducted a detailed study of the architecture of XTS-300 during which they analyzed the documentation and source code in detail.

The architecture study yielded several minor flaws. They were corrected by Wang. Penetration testing uncovered one implementation flaw, which was fixed by Wang. The testing performed by the team showed that the system was resistant to penetration. The team found that the TCB implementation was consistent with the descriptive top-level specification. No design flaws were found and the implementation flaws that were found during testing were corrected. The team is confident that the system is resistant to penetration.

Conclusion

XTS-300 satisfies the B3 Security Testing requirement.

8.21 Design Specification and Verification

Requirement

A formal model of the security policy supported by the TCB shall be maintained that is proven consistent with its axioms. A descriptive top-level specification (DTLS) of the TCB shall be maintained that completely and accurately describes the TCB in terms of exceptions, error messages, and effects. It shall be shown to be an accurate description of the TCB interface. A convincing argument shall be given that the DTLS is consistent with the model.

Applicable Features

The vendor has provided a security model interpretation document [24] based on the Bell and LaPadula model with the Biba model incorporated through a combined dominance relation and the Multics Interpretation.

The vendor has provided a DTLS [16] that, in combination with the Trusted Programmer's Reference Manual [34] and the User's Manual [35], describes the TCB operations through the kernel gates, the TSS gates, and the trusted commands.

The team performed a manual review and comparison analysis of the DTLS and the TCB Detailed Specifications [21, 22, 23] and found them to be consistent. The Detailed Specifications describe the TCB software at the pseudo-code level.

The team performed a manual review and comparison analysis of the model and the DTLS, the TPRM, and the User's Manual and found the latter three documents to be consistent with the model.

Conclusion

XTS-300 satisfies the B3 Design Specification and Verification requirement.

8.22 Configuration Management

Requirement

During development and maintenance of the TCB, a configuration management system shall be in place that maintains control of changes to the descriptive top-level specification, other design data, implementation documentation, source code, the running version of the object code, and test fixtures and documentation. The configuration management system shall assure a consistent mapping among all documentation and code associated with the current version of the TCB. Tools shall be provided for generation of a new version of the TCB from source code. Also available shall be tools for comparing a newly generated version with the previous TCB version in order to ascertain that only the intended changes have been made in the code that will actually be used as the new version of the TCB.

Applicable Features

The vendor's configuration management plan was described earlier in this report (see Section 7.5, page 109). It describes the techniques used to maintain control of changes to the system. All design documentation, including the DTLS, and all source code, including test fixtures, are maintained under SCCS. The diff command is used to compare newly generated versions with previous versions to ensure that all intended changes are incorporated in the new version.

Conclusion

XTS-300 satisfies the B3 Configuration Management requirement.

8.23 Security Features User's Guide

Requirement

A single summary, chapter, or manual in user documentation shall describe the protection mechanisms provided by the TCB, guidelines on their use, and how they interact with one another.

Applicable Features

The vendor has provided a User's Manual [35]. The document describes an overview of the system. The document also includes descriptions of how to use the trusted path, how to login, password control, and a description of all of the trusted commands available to the user. The User's Manual also contains the shell commands.

Conclusion

XTS-300 satisfies the B3 Security Features User's Guide requirement.

8.24 Trusted Facility Manual

Requirement

A manual addressed to the ADP system administrator shall present cautions about functions and privileges that should be controlled when running a secure facility. The procedures for examining and maintaining the audit files as well as the detailed audit record structure for each type of audit event shall be given. The manual shall describe the operator and administrator functions related to security, to include changing the security characteristics of a user. It shall provide guidelines on the consistent and effective use of the protection features of the system, how they interact, how to securely generate a new TCB, and facility procedures, warnings, and privileges that need to be controlled in order to operate the facility in a secure manner. The TCB modules that contain the reference validation mechanism shall be identified. The procedures for secure generation of a new TCB from source after modification of any modules in the TCB shall be described. It shall include the procedures to ensure that the system is initially started in a secure manner. Procedures shall also be included to resume secure system operation after any lapse in system operation.

Applicable Features

The vendor has provided two manuals for administrators and operators. The Trusted Facility Manual [33] is geared to the security administrator for the system and contains an overview of the system, a comprehensive description of the system's security mechanisms, guidelines on maintaining a secure system, and a description of all the trusted commands available to all administrators and operators. It describes the TCB, indicating what portions of the TCB contain the reference validation mechanism. The document also contains descriptions of the audit records for each of the auditable events. It explains how to install the TCB initially, and how to bring it into operation, both from a normal start, and following a system failure. Another manual, providing information for programmers, is called "Trusted Programmer's Reference Manual" [34]. It contains a description of all visible TCB gates, and a description of all routines in the trusted software library. There are no provisions for a site's generating a new TCB from source; there is only a single version, and it is generated by the vendor.

Conclusion

XTS-300 satisfies the B3 Trusted Facility Manual requirement.

8.25 Test Documentation

Requirement

The system developer shall provide to the evaluators a document that describes the test plan, test procedures that show how the security mechanisms were tested, and results of the security mechanisms' functional testing. It shall include results of testing the effectiveness of the methods used to reduce covert channel bandwidths.

Applicable Features

Wang has supplied the team with a set of documents [26, 27, 28, 29, 17] that describe a comprehensive approach toward testing the XTS-300 and its security mechanisms. The test documentation includes a test plan, a test user's guide, and a two-volume test procedure document. Wang's test suite tests all user-visible error returns from the TCB gates, as well as the designed functionality. The test coverage analysis document traces parameter passing for all of the gates that call service-level access-decision functions inside the kernel and TSS, as well as from TSS to the kernel for those TSS functions that directly pass access information to the kernel.

The methods and effectiveness of reducing covert channel bandwidths are described and analyzed in Wang's Covert Channel Analysis (CCA) [20].

Conclusion

XTS-300 satisfies the B3 Test Documentation requirement.

8.26 Design Documentation

Requirement

Documentation shall be available that provides a description of the manufacturer's philosophy of protection and an explanation of how this philosophy is translated into the TCB. The interfaces between the TCB modules shall be described. A formal description of the security policy model enforced by the TCB shall be available and proven that it is sufficient to enforce the security policy. The specific TCB protection mechanisms shall be identified and an explanation given to show that they satisfy the model. The descriptive top-level specification (DTLS) shall be shown to be an accurate description of the TCB interface. Documentation shall describe how the TCB implements the reference monitor concept and give an explanation why it is tamper resistant, cannot be bypassed, and is correctly implemented. The TCB implementation (i.e., in hardware, firmware, and software) shall be informally shown to be consistent with the DTLS. The elements of the DTLS shall be shown, using informal techniques, to correspond to the elements of the TCB. Documentation shall describe how the TCB is structured to facilitate testing and to enforce least privilege. This documentation shall also present the results of the covert channel analysis and the tradeoffs involved in restricting the channels. All auditable events that may be used in the exploitation of known covert storage channels shall be identified. The bandwidths of known covert storage channels, the use of which is not detectable by the auditing mechanism, shall be provided.

Applicable Features

The vendor has provided the hardware documentation for the Intel Pentium II/III [9, 6], for the peripherals [1, 10], and for hardware integrity [17].

The System Architecture document [25] and the DTLS [16] provide the philosophy of protection, and explain how this philosophy is translated into the TCB.

The System Architecture document [25] defines the interfaces between the TCB modules. The Detailed Software Specifications [21, 22, 23] describe the module interfaces in detail (the Detailed Specifications describe the software implementation down to the pseudo-code level). The Bell and LaPadula model [4] with the Biba model is used as the abstract model and the Multics Interpretation [4] is used as the concrete model. The vendor has also provided the Security Model Interpretation [24] relating the state-transitioning TCB interfaces and the corresponding rules to the basic 11 rules in the Multics Interpretation.

The team has carried out a manual review and analysis of the Security Model Interpretation, the DTLS, and the Detailed Software Specifications. These documents are consistent with each other. Thus, the team concludes that the DTLS is an accurate description of the TCB interface, the DTLS and implementation are consistent with each other, and the elements of the DTLS correspond to the elements of the TCB.

The System Architecture document presents the argument that the kernel implements the reference monitor concept and that the kernel is tamper-resistant and cannot be bypassed. In addition to the consistency

among the Security Model, the DTLS, and the Detailed Software Specifications, testing will provide further assurance that the kernel is correctly implemented.

The System Architecture document describes the TCB structure and presents a convincing argument that the principle of least privilege is enforced. After reviewing the TCB structure described in the System Architecture document, the team concluded that the TCB structure was designed to facilitate easy and comprehensive testing.

The vendor provided a Covert Channel Analysis document [20] that identifies the storage and timing channels. The document also contains the computations of the bandwidths of these channels. The document further identifies the resource exhaustion events that must be audited to detect the potential exploitation of the storage channels. Finally, the document provides a mechanism (time delays) to reduce the bandwidths of the storage channels caused by resource exhaustions.

Conclusion

XTS-300 satisfies the B3 Design Documentation requirement.

This page intentionally left blank

Chapter 9

Evaluator Comments

This section of the report contains some of the more personal comments of the evaluation team, particularly those that deal with aspects of the system that do not directly affect the rating assigned. Some of the comments expressed here may be of value to prospective users of the system and also to those responsible for determining the suitability of this system in a particular application.

9.1 Knowledgeable, Cooperative Vendor

The advantages of working with an experienced, conscientious vendor should not be overlooked. Wang has shown over and over again its dedication to providing a solid system that meets the TCSEC B3 requirement. It worked closely and supportively with the team. The interaction with the vendor was exemplary, and helped make this evaluation move more smoothly than it would have otherwise. Wang repeatedly brought up issues that might reasonably be expected to be missed otherwise, even when deadlines were imminent and without apparent regard to whether fixes might be easy or difficult. This reflects Wang's dedication to producing a secure product and gave the NSA part of the team added confidence in the quality of the RAMP and the security of XTS-300.

9.2 Technical Advantages

There are some positive technical aspects of the system that the team felt should be called to the reader's attention. They are:

- The command and programming interface implementation is very close to standard UNIX
- The file system is contained in the TCB, so that the TCB actually implements the protected objects as opposed to an unusual and unique emulation layer
- The system uses a central, simple mechanism (mandatory integrity) to enforce administrative policies
- Users can optionally employ the mandatory integrity policy to protect files and programs from tampering.

9.3 The tdc Utility

Wang provides a tdc command which is used to verify that the product sent to the customer site is the one received. The tdc command is executed by Wang on the release media before the system is shipped using a

release distribution key. A 16-bit Cyclic Redundancy Check (CRC) is used and the checksum is encrypted. The release media, distribution key, and resulting checksum are shipped separately to the customer site. It is intended that the customer then apply the tdc command to the release media and compare the checksum generated to the one that was shipped. The tdc command, while it provides some measure of assurance concerning the delivered system, does not provide adequate assurance to satisfy the trusted distribution A1 level requirement.

9.4 User Interface

The team has some concerns about the clumsiness of the user interface exhibited by the Trusted Commands. While this may arise inevitably from the security requirements, some team members feel the interface could be improved. Some of the problems the team found are:

- Only a single command, request, or parameter can be entered on a line, with no provisions for type-ahead.
- Some commands seem unnecessarily verbose in their output; for example, the audit and param_edit commands list all possible audit events under some circumstances when it is deemed unnecessary, but there is no way to avoid the several screens of output.

9.5 Need for Unevaluated Applications or CASS

The evaluated system has no direct user interface, other than through the trusted processes. While this reduces the complexity of the evaluation, it does place a burden on the site administration. It is clear that a site will, of necessity, use CASS, or provide some unevaluated software as the user interface.

9.6 Password Checking

The system offers no automatic password generation schemes, nor does it check for weak passwords chosen by the user. However, the system does check for a minimum password length of six characters and a maximum length of 15 characters.

9.7 Limited Number of ACLs

A user is limited to a maximum of seven distinct ACL entries that can be used to designate an individual user or a group. This seems to be a restrictively small number given the potential number of customers for a system such as this, some of whom may wish to employ the system with a large number of users.

9.8 One System Sync Interval

The amount of audit data that is subject to loss in the event of a system failure depends on the length of the system sync interval. This sync interval determines how frequently disk blocks are written to disk. If the value is small, system activity increases and performance may be adversely affected. The team suggested to Wang that two values be used for the system sync process: one value would apply to audit segments while the other would apply to the rest of the system. The team recommended to the vendor that this feature be incorporated into future versions of the system.

9.9 Audit

Experience during testing has shown that the audit system was designed to just meet the requirements. There are difficulties in effectively using the audit system to analyze information. An off-line analysis tool and more high-level information in the audit trail would be useful, but are not provided by the vendor in the evaluated configuration, nor is there any way (using the TCB) to capture the audit information in human readable form.

This page intentionally left blank

Appendix A
Evaluated Hardware Components

Wang Government Services, Incorporated markets the evaluated system as XTS-300. The hardware components are given individual identifiers by the manufacturer. Rather than list all the available components, this appendix summarizes the hardware.

A.1 The Intel Pentium II/III Motherboard

The motherboard used on the Intel Pentium II/III hardware base is the Intel R440LX (LX) or the L440GX+/L440GXC (GX) and contains the following components:

- Intel 300 MHz Pentium II or 500 MHz Pentium III
- Intel 82443LX (LX) or 82443GX (GX) PCI/AGP Controller (PAC)
- Intel 82371AB (LX) or 82371EB (GX) PCI-ISA Bridge (PIIX4)
- National Semiconductor PC87307 (LX) or PC87309 (GX) Super I/O
- Adaptec AIC-7880 (LX) or AIC-7896 (GX) SCSI Host Adapter
- Cirrus Logic CL-GD5446 (LX) or CL-GD5480 Video Controller
- Intel NIC chip
- BIOS
- 64 - 512 Mbyte RAM

A.2 Additional Controller Boards

- Optional Applied Digital Four-Port Interface SIO4 Controller Card
- Optional Znyx two (ZX348/ZX348Q) or four (ZX346/ZX346Q) port NetBlaster LAN Adapter
- Optional add-in Adaptec AHA-2940U or AHA-2940U2W SCSI Host Adapter

A.3 Generic Devices

The items on this list have been specified in a generic manner.

- Generic 101-key standard, 104-key standard or laptop-style Keyboard
- VGA monitor or flat panel display
- 3-button, PS/2 or serial mouse/touchpad
- terminal (see below)
- dumb printer (see below)

A.4 Terminals

Those terminals used in an evaluated system must have certain characteristics. These characteristics are defined in the Trusted Facility Manual (TFM) and summarized below:

- Provide a keyboard and a video display.
- Provide a correct association between the user's actions of depressing keyboard keys and transmitting the ASCII representations of those keys via the communications interface.
- Provide a direct means of generating an out-of-band BREAK signal that can be used to initiate the trusted path for communicating directly with the TCB. This BREAK signal is defined by the serial communication interface.
  - It must not be able to generate the BREAK signal in response to any sequence of signals sent from the system.
  - Its ability to generate a BREAK signal cannot be inhibited by any sequence of signals sent from the system. The same conditions should exist if this BREAK signal were an in-band ASCII character or sequence of characters.
- Provide a means to set and clear the Data Terminal Ready (DTR) signal and the Carrier Detect signal to indicate the presence or absence of a properly working terminal attached to the TCB's hardware.
  - The DTR and Carrier Detect signals are defined by the serial communication interface.
  - The DTR and Carrier Detect signals can be provided by the terminal itself (power on or off signal) or by an external mechanism interposed between the terminal and the TCB's hardware interface.
- Provide a correct association between the ASCII characters transmitted from the system and their displayed or printed human-readable printed representations.
- Provide the means for the terminal user to clear any internal or external storage associated with the terminal between user sessions. This includes clearing the screen display, programmable function keys, buffer memory, printer paper, and local disk. It is the responsibility of the user or local computer security administrative personnel to ensure that this action be exercised between user sessions.

If it is possible to program (either under user or system control) any characteristics of the terminal connected to the TCB (such as correspondence between keystrokes and transmitted characters).

- The means for doing so must be fully documented and available to the user.
- It must be possible for the terminal user to return all programmable characteristics to a default or well-known state.
- It is the user's responsibility to ensure that the terminal is appropriately reset to known characteristics any time the user invokes the trusted path if the terminal's programmable characteristics can be set by untrusted code. It is the responsibility of the user or local computer security administrative personnel to ensure that appropriate procedures for doing this be defined.

There is additional guidance in the TFM concerning terminals with advanced features. An example of such a terminal is a terminal of type VT100 or VIP7800. No terminal emulators are supported in the evaluated configuration.

A.5 Printers

The following describes the generic, dumb printers that are supported as system printers connected to asynchronous communication lines:

Any parallel or any serial printer that uses XON/XOFF for flow control, it must accept printable ASCII data and does not require special initialization sequences. Acceptable ASCII data includes the ASCII character set and control characters, such as <CR>, <LF>, null, XON/XOFF, and excludes printers that have special graphics modes such as postscript printers. An example of such a printer is the Epson FX80 dot matrix printer with a serial interface. As long as the printer does not support Postscript or any other interpretive language (with the exception of the PCL-5 printer mentioned below), it can be included in the evaluated configuration.

A single PCL-5 printer is also available in the evaluated configuration: the Hewlett-Packard 4000. Special administrative procedures are required when a PCL-5 printer is included in the configuration.

A.6 Additional Devices

- IBM Ultrastar SCSI hard disk drives (4.5 Gbyte, 9.1 Gbyte, 18.3 Gbyte, and 36.7 Gbyte) Models 9ES, 18ES, 9ZX, 18ZX, 9LZX, 18LZX, 36LZX, 9LP, 18XP, and 36XP.
- WangDAT 3400DX 4mm DAT SCSI tape drive
- Hewlett Packard 1554A 4mm DAT SCSI tape drive
- Toshiba XM-6401B (tray loading) internal SCSI CD-ROM drive
- Spyrus RD400S2C1I SCSI PC Card reader, internal, dual-slot (MS-MCDISK-E-1)
- Litronic 2108 SCSI PC card reader, external eight-slot (CipherServer)
- Fairchild Defense Data Transfer Cartridge Reader

A.7 Acceptable Hardware Changes

In recognition of the rapid pace at which hardware components are revised, a category of insignificant hardware changes has been established. Changes categorized as insignificant are regarded as having no affect on the security policy of the system or the B3 rating. That is, hardware component revisions are considered to be acceptable in the evaluated configuration, if the changes fall into the category of changes listed below. The specific components listed above serve to document those specifically investigated by the evaluation team. Site personnel who wish to verify that the changes to hardware components fall into the category listed below should contact Wang. Any revision to a hardware component that incorporates changes beyond those listed below is not acceptable in the evaluated configuration. Use of hardware components that contain such revisions invalidates the B3 rating for the XTS-300 system.

The following is a list of acceptable changes to hardware components:

- Any changes to the system power supply, cooling fans, case/card cage, or cords;
- Changes in the fastening mechanism, arrangement, or orientation of any component or subcomponent;
- Changes to the color, weight, size, shape of any component or subcomponent;
- Any change to "non-chip" board components (such as resisters and capacitors) or to the number of such components;
- Changes to the voltage requirements/tolerances or electrical signal profiles input or output by any component or subcomponent.

Appendix B

Evaluated Software Components

The following are the components of STOP 5.2.E that comprise the evaluated the Trusted Computing Base (TCB):

- Security Kernel
- TCB System Services (TSS)
- Trusted Software

The Commodity Application System Services (CASS), although a portion of STOP 5.2.E shipped with the system, are not part of the TCB and were not evaluated.

This page intentionally left blank

Appendix C

Draft Evaluated Products List Entry

DATE:	30 April 2000
MAINTAINED PRODUCT:	XTS-300 Release 5.2.E
PREVIOUS PRODUCTS:	XTS-300 Release 4.4.2 XTS-300 Release 4.1a XTS-300 Release 4.1 XTS-200 Release 3.2.E XTS-200 Release 3.1.E
VENDOR:	Wang Government Services, Inc.
EVALUATION CLASS:	B3

C.1 Product Description

The XTS-300 product is a combination of STOP 5.2.E, a multilevel secure operating system, and a Wang supplied x86 hardware base. (STOP is not licensed to run on non-Wang supplied hardware and would lose its evaluated status if that was attempted.) STOP is a multiprogramming system that can support terminal connections for up to 19 users. Up to 200 processes can run concurrently, each with up to four gigabytes of virtual memory. STOP is designed not only to support much of the UNIX System V interface for applications software, but to produce and run object programs that adhere to a subset of the "Intel386 Family Binary Compatibility Specification 2" as well.

An X-windows graphical user interface (GUI) is supported by the TCB. It is available at the console for work by untrusted users. Trusted path initiation causes suspension of the GUI and trusted commands can not be run from the GUI. All windows on the display are at the same level and multi-level cut-and-paste is not supported.

Network connectivity is allowed in the evaluated configuration. TCP/IP and Ethernet are built in to the Trusted Computing Base, but no network servers (e.g., SMTP) are within the TCB. Within an evaluated configuration, network attachments must be made according to rules in the Trusted Facility Manual (e.g., the network must be single-level while multiple networks can each be at a different level). The TCB can not be compromised by remote users or unusual network traffic, but the TCB itself, like most systems, does not prevent disclosure of, or loss of integrity by, data on the network.

STOP consists of four components: the Security Kernel, which operates in the most privileged ring and provides all mandatory, subtype, and a portion of the discretionary, access control; the TCB System Services, which operate in the next-most-privileged ring, and implements a hierarchical file system, supports user I/O,

and implements the remaining discretionary access control; Trusted Software, which provides the remaining security services and user commands; and Commodity Application System Services (CASS), which operate in a less privileged ring and provides the UNIX-like interface. CASS is not in the TCB.

The XTS-300 is hosted on Intel Pentium II or Pentium III based server class systems, available in tower, rack-mount and tempest form factors. The XTS-300 uses specific Intel-brand motherboards and industry standard ISA and PCI peripheral cards or chips built into the motherboard. The XTS-300 product is a combination of the STOP 5.2.E multilevel secure operating system and a Wang supplied hardware base. (STOP is not licensed to run on non-Wang supplied hardware and the system would lose its evaluated status if that was attempted.)

STOP provides support for peripherals such as:

- Ultra2Wide hard disks (4.5GB to 36.4GB)
- CD ROMs
- PCMCIA/PC-Card readers supporting Fortezza encryption devices
- Tape drives to 24GB
- SVGA video including UNIX's X-Windows
- Floppy disk
- TCP/IP based 10 or 100BaseT networks
- Fairchild-proprietary data transfer cartridges (DTCR)
- Serial or parallel printers
- Mouse or Touchpad
- Keyboard

The system provides mandatory access control (MAC) that allows for both a security and integrity policy. The mandatory security policy enforced by the XTS-300 is based on the Bell and LaPadula security model; the mandatory integrity policy is based on the Biba integrity model. The system implements discretionary access control (DAC) and provides for user identification and authentication needed for user ID-based policy enforcement. The system also provides an additional policy mechanism, "subtypes," which is not required by the Trusted Computer System Evaluation Criteria and which can be used in a customer-specific way in conjunction with MAC and DAC controls.

Individual accountability is provided with an auditing capability. Data scavenging is prevented through object reuse prevention mechanisms. A trusted path mechanism is provided by the implementation of a Secure Attention Key (SAK).

The separation of administrator and operator roles is enforced using the integrity policy. The system enforces the "principle of least privilege" (i.e., users should have no more authorization than that required to perform their functions) for administrator and operator roles. All actions performed by privileged (and normal) users can be audited. The audit log is protected from modification using integrity and subtype mechanisms. STOP also provides an alarm mechanism to detect the accumulation of events that indicate an imminent violation of the security policy.

The TCB exhibits strong architectural characteristics: minimization, layering, abstraction, and data hiding. The TCB makes use of hardware features to provide process separation and TCB isolation and has been designed and implemented to resist penetration. The system design is based on a security model and a descriptive top-level specification.

C.2 Product Status

The STOP operating system was developed by (and is marketed and supported by) Wang Government Services, Inc. (hereafter referred to as Wang). Release 5.2.E of the XTS-300 can be ordered after May 1, 2000. Orders can be placed with:

Mike Focke
XTS Product Manager
Wang Government Services, Inc.
7900 Westpark Drive
McLean, Virginia 22102
(703) 464–8754
Internet: mike.focke@wang.com

C.3 Product Changes Since Previous Evaluation/RAMP Action

The following major enhancements have been added to the TCB since the last RAMP:

- Support for the Pentium-II and Pentium-III platforms
- Multi-processor support (up to two 500MHz Pentium-III processors)
- Support for synchronous SCSI access
- Support for SCSI speeds/connections through Ultra2Wide (80Mb/sec)
- Support for large disks up to 36.4GB
- Litronic Cipher-Server
- Support for Parallel laser printers
- Support for 100BaseT network cards
- Support for a Fast File system
- Support for new video controllers
- Support for a larger (24GB) tape drive
- Up to 862MB of main memory is now allowed

C.4 Security Evaluation Summary

The security protection provided by the original product, XTS-200 release 3.1.E, was evaluated by the National Security Agency (NSA) against the requirements specified by the Department of Defense Trusted Computer System Evaluation Criteria [DOD 5200.28-STD] dated December 1985. The NSA evaluation team determined that the system satisfied all the specified requirements of the Criteria at class B3. The original evaluation was completed in May 1992. The original product was produced by Honeywell Federal Systems Inc. (HFSI), which has since been acquired by Wang.

XTS-300 release 5.2.E is based on release 4.4.2 of the XTS-300, which was in turn based on release 4.1a of the XTS-300. Release 4.1a successfully completed a Ratings Maintenance Phase (RAMP) action in October 1995. Release 4.4.2 successfully completed a RAMP action in March 1998.

To RAMP release 5.2.E of the XTS-300, the vendor maintained the security properties of release 4.4.2, performed configuration management of the changes, and enhanced the security test suite and the system documentation appropriately. The security protection provided by XTS-300 release 5.2.E has been evaluated against the requirements specified in the Criteria by a joint NSA/vendor security analysis team (SA-team). The SA-team has also evaluated the system change procedures followed by the vendor against the B2+ RAMP requirements in the RAMP Program Document dated March 1995.

The SA-team has determined that release 5.2.E of the XTS-300 satisfies all the specified requirements of the Criteria at class B3 and that the vendor satisfied all the B2+ RAMP requirements. The conclusions of the SA-team have been reviewed and approved by NSA. For a complete description of how XTS-300 release 5.2.E satisfies each requirement of the Criteria, see Final Evaluation Report, Wang Government Services, Inc., XTS-300 (Report CSC-EPL-92/003.E).

The Final Evaluation Report should be consulted for the complete lists of evaluated hardware and software components.

C.5 Environmental Strengths

The XTS-300 is a general-purpose computer system. The B3 rating implies not only incorporation of particular security features, but a very high level of assurance. This level of assurance should allow the XTS-300 to be accredited to handle data at a wide range of classification levels in a wide range of environments. Several certification and accreditation efforts have been completed that use the XTS-300 as a multi-level application platform.

The XTS-300 is general-purpose in that it can be used for a range of purposes from multi-user workstation, rack-mount and tempest variants to guard/gateway. With additional application support, it is suitable as a network server or firewall. Since the XTS-300 is based on commodity hardware, it is positioned to take advantage of the frequent hardware advances in the x86 hardware base and in the SCSI subsystem.

Beyond the minimal requirements for a B3 system, the XTS-300 provides a mandatory integrity policy, an extra subtype policy, a familiar, Unix-like environment for single-level applications, and secure distribution tools. Integrity can be used for, among other things, virus protection. The Unix-like environment supports binary compatibility and will run many programs imported from other systems without recompilation or with minor porting.

Appendix D
Acronyms

AC	Alignment Check
ACK	Acknowledgement
ACL	Access Control List
ADDR	Address
ADP	Automated Data Processing
ADT	Active Device Table
ADTE	Active Device Table Entry
AF	Auxiliary Carry Flag
AM	Associative Memory; Alignment Mask
ANSI	American National Standards Institute
APT	Active Process Table
APIC	Advanced Programmable Interval Controller
APTE	Active Process Table Entry
ASCII	American Standard Code for Information Interchange
AST	Active Segment Table
ASTE	Active Segment Table Entry
AT&T	American Telephone and Telegraph
BCS	Binary Compatibility Standard
bps	Bits per second
BIOS	Basic Input/Output System
CAM	Content Addressable Memory (associative cache)
CASS	Commodity Application System Services
CCA	Covert Channel Analysis
CCB	Configuration Control Board
CD	Cache Disable
CF	Carry Flag
CLE	Connection List Entry
COTS	Commercial-off-the-shelf
CPL	Current Privilege Level
CPU	Central Processing Unit
CR	Carriage Return
CS	Current Segment; Code Segment
CTE	Channel Table Entry
CUP	Change User Password
DAC	Discretionary Access Control
DAP	Design Analysis Phase
DBC	Data Bus Controller
DBTE	Device Branch Table Entry
DCE	Data Circuit-Terminating Equipment
DCTE	Device Control Table Entry
DD	Duplicate Directory
DES	Data Encryption Standard
DF	Direction Flag
DMA	Direct Memory Addressing
DMA/R	DMA/Refresh Controller
DOD	Department of Defense
DPA	Dataproduct Printer Adapter
DPL	Descriptor Privilege Level

DS	Data Segment
DTE	Device Table Entry
DTLS	Descriptive Top-Level Specification
E-CACHE	Data Cache
EBC	EISA Bus Controller
EEPROM	Electrically Erasable Programmable Read-Only Memory
EIP	Instruction Pointer Register
EISA	Extended Industry Standard Architecture
EM	Emulation
EPL	Evaluated Products List
ES	Data Segment
FCA	Filesystem Control Area
FD	File Descriptor
FIFO	First-In First-Out
FPU	Floating Point Unit
FS	File System; Data Segment
FTE	File Table Entry
G	Granularity
Gbyte	Gigabyte (2^{30} bytes)
GDTR	Global Descriptor Table Register
GLINK	Terminal Emulation Package
GS	Data Segment
HFS	(Not an acronym. Formerly: Honeywell Federal Systems)
HFSI	HFS Incorporated
I-CACHE	Instruction Cache
I/O	Input/Output
I/OAT43	Serial/Parallel Adapter Card
ID	Identification/Identifier
IDT	Interrupt Descriptor Table
IDTR	Interrupt Descriptor Table Register
IF	Interrupt-enable Flag
IEEE	IEEE
IO	Input/Output
IOPL	I/O Privilege Level
IOQE	I/O Queue Entry
IP	Internet Protocol
IPAR	Initial Product Assessment Report
IPC	Interprocess Communication
IRQ	Interrupt Request
ISA	Industry Standard Architecture
ISC	Intelligent SCSI Controller
ISMn	Interrupt Save Mask n
ISN	Internal Software Note
ISO	International Organization for Standards
ISP	Integrated System Peripheral
Kbyte	Kilobyte (2^{10} bytes)
KC	Keyboard Controller
KDT	Known Device Table
KDTE	Known Device Table Entry
KST	Known Segment Table
KSTE	Known Segment Table Entry
LDD	Logical Device Data
LDTR	Local Descriptor Table Register
LF	Line Feed

MAC	Mandatory Access Control
Mbyte	Megabyte (2^{20} bytes)
MCC	Memory/Cache Controller
MP	Math Present
MSB	Most Significant Bit
MSG	Message Signal
MTE	Mount Table Entry
NAK	Negative Acknowledgement
NATO	North Atlantic Treaty Organization
NCSC	National Computer Security Center
NE	Numeric Error
NMI	Non-Maskable Interrupt
NSA	National Security Agency
NT	Nested Task
NVM	Non-Volatile Memory
NW	Not Write-through
OF	Overflow Flag
OSS	Operating System Services
PADT	Previously Active Device Table
PC	Program Counter (Hardware); IBM Personal Computer or compatible (Testing)
PCD	Page-level Cache Disable
PD	Page Descriptor
PDIR	Paging Directory
PDS	Process Descriptor Segment
PE	Protection Enable
PF	Parity Flag
PG	Paging Bit
PIC	Programmable Interrupt Controller
PID	Process ID
PIT	Programmable Interval Timer
PLDS	Process Local Data Segment
POSIX	Portable Operating System Interface for Computer Systems
POST	Power-On and System Reset
PRA	Line Printer Adapter
PROM	Programmable Read Only Memory
PT	Process Type
PTE	Page Table Entry
PTR	Preliminary Technical Review
PWT	Page-level Writes Transparent
QIC	Quarter-Inch Cartridge
R	Read Indicator
RAM	Random Access Memory
RAMP	Rating Maintenance Phase
RF	Resume Flag
RN	Ring Number
ROM	Read Only Memory
RPL	Requestor's Privilege Level
RST	Reset
RTC	Real Time Clock
SA	System Arbiter
SAK	Secure Attention Key
SBTE	Segment Branch Table Entry
SCCS	Source Code Control System (UNIX utility)
SCOMP	Secure Communications Processor

SCSI	Small Computer Systems Interface
SD	Segment Descriptor
SF	Sign Flag
SFUG	Security Features User's Guide
SG	Set Group
SL	Set Level
SS	Stack Segment
SSEG	Secure Systems Engineering Group
STD	Standard
STOP	(not an acronym)
SW	Software
Tbyte	Terabyte (2^{40} bytes)
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TCSEC	Trusted Computer System Evaluation Criteria
TDC	Time of Day Clock
TF	Trap Flag
TFM	Trusted Facility Manual
TI	Table Indicator
TR	Task Register
TRB	Technical Review Board
TPRM	Trusted Programmer's Reference Manual
TS	Task Switched
TSC	Time Slice Clock
TSKSS	Task State Segment
TSS	TCB System Services
U/S	User/Supervisor
UART	Universal Asynchronous Receiver and Transmitter
uid	Unique Identifier
VA	Virtual Address
VAP	Vendor Assistance Phase
VM	Virtual 8086 Mode
WP	Write Protect
ZF	Zero Flag

This page intentionally left blank

Appendix E

Bibliography and References

[1] Addison-Wesley. *System BIOS for IBM PCs, Compatibles, and EISA Computers: The Complete Guide to ROM-Based System Software*. Reading, MA, 1991.

[2] American National Standards Institute: X3, Information Processing Systems. *American National Standards for Information Systems: Programming Language C*. Washington, D.C., February 1990. X3J11/90-013.

[3] American Telephone and Telegraph (AT&T). *AT&T System V Interface Definition, Issue 2*, 1986. (three volumes).

[4] Bell, D. E., and LaPadula, L. J. Computer Security Model: Unified Exposition and Multics Interpretation. Technical Report ESD-TR-75-306, The MITRE Corporation, Bedford, MA, June 1975.

[5] Biba, K. J. Integrity Considerations for Secure Computer Systems. Technical Report ESD-TR-76-372, The MITRE Corporation, Bedford, MA, April 1977.

[6] Crawford & Gelsinger. *Programming the 80386*. Sybex, Alameda, CA, 1987.

[7] Institute of Electrical and Electronics Engineers. *ISO/IEC 9945-1: 1990 (IEEE Std 1003.1990) Information Technology-Portable Operating System Interface (POSIX) – Part 1: System ApplicationProgram Interface (API)*. New York, NY, 1990. IEEE Standard 1003.1990.

[8] Intel Corporation. *Intel386 Family Binary Compatibility Specification 2*. Santa Clara, CA. Order Number 468366-001.

[9] Intel Corporation. *Intel 486 Microprocessor Family Programmer's Reference Manual*. Mt. Prospect, IL, 1992.

[10] Megatrends. *ISA and EISA: Hi-Flex AMIBIOS Technical Reference*. Norcross, GA, 1993.

[11] National Computer Security Center. *Department of Defense Trusted Computer System Evaluation Criteria*. Linthicum, MD, December 1985. DoD 5200.28-STD.

[12] National Computer Security Center. *Final Evaluation Report: Honeywell Multics MR11.0*. Linthicum, MD, June 1985. CSC-EPL-85/003.

[13] National Computer Security Center. *Final Evaluation Report: Secure Communications Processor (SCOMP) STOP Release 2.1*. Linthicum, MD, September 1985. CSC-EPL-85/001.

[14] National Computer Security Center. *Final Evaluation Report: HFS Incorporated XTS-200*. Linthicum, MD, January 1994. CSC-EPL-92/003.A.

[15] Wang Federal, Inc. *STOP Release 3.1, TCB Test Coverage Analysis, XTS-300*. McLean, VA, June 1991. Document No. FS91-511, Rev. 2.

[16] Wang Federal, Inc. *STOP Release 4.1, Descriptive Top-Level Specification, XTS-300.* McLean, VA, 1994. Document No. FS94-275-00.

[17] Wang Federal, Inc. *XTS-300 System Integrity Installation and User's Manual.* McLean, VA, April 1995. Document No. FS92-377-03.

[18] Wang Government Services, Inc. *C Coding Standards TCB.* McLean, VA, June 1999. Version 2.0.

[19] Wang Government Services, Inc. *Rating Maintenance Plan XTS-200/XTS-300.* McLean, VA, April 2000. Document No. FS92-361.

[20] Wang Government Services, Inc. *STOP Release 5.2.E, Covert Channel Analysis XTS-300.* McLean, VA, April 2000.

[21] Wang Government Services, Inc. *STOP Release 5.2.E, Detail Specification, Security Kernel Software.* McLean, VA, April 2000. Document No. FS94-278.

[22] Wang Government Services, Inc. *STOP Release 5.2.E, Detail Specification, TCB System Services.* McLean, VA, April 2000. Document No. FS94-279.

[23] Wang Government Services, Inc. *STOP Release 5.2.E, Detail Specification, Trusted Software.* McLean, VA, April 2000. Document No. FS94-280.

[24] Wang Government Services, Inc. *STOP Release 5.2.E, Security Model Interpretation, XTS-300.* McLean, VA, April 2000. Document No. FS94-281.

[25] Wang Government Services, Inc. *STOP Release 5.2.E, System Architecture, XTS-300.* McLean, VA, April 2000. Document No. FS94-276.

[26] Wang Government Services, Inc. *STOP Release 5.2.E, TCB Test Plan, XTS-300.* McLean, VA, April 2000. Document No. FS94-284.

[27] Wang Government Services, Inc. *STOP Release 5.2.E, TCB Test Procedures Vol. 1, XTS-300.* McLean, VA, April 2000. Document No. FS94-284.

[28] Wang Government Services, Inc. *STOP Release 5.2.E, TCB Test Procedures Vol. 2, XTS-300.* McLean, VA, April 2000. Document No. FS94-284.

[29] Wang Government Services, Inc. *STOP Release 5.2.E, TCB Test User's Guide, XTS-300.* McLean, VA, April 2000. Document No. FS94-283.

[30] Wang Government Services, Inc. *XTS-200/XTS-300 Configuration Management Plan.* McLean, VA, May 2000. Document No. FS93-161.

[31] Wang Government Services, Inc. *XTS-300 Application's Programmer's Reference Manual.* McLean, VA, April 2000. Document No. FS92-374.

[32] Wang Government Services, Inc. *XTS-300 STOP Release 5.2.E Software Release Bulletin.* McLean, VA, April 2000. Document No. FB90-129.

[33] Wang Government Services, Inc. *XTS-300 Trusted Facility Manual.* McLean, VA, April 2000. Document No. FS92-371.

[34] WANG GOVERNMENT SERVICES, INC. *XTS-300 Trusted Programmer's Reference Manual.* McLean, VA, April 2000. Document No. FS92-375-02.

[35] WANG GOVERNMENT SERVICES, INC. *XTS-300 User's Manual.* McLean, VA, April 2000. Document No. FS92-373.

[36] WRIGHT, G. R., AND STEPHENS, W. R. *TCP/IP Illustrated Volume 2.* Addison-Wesley Publishing Company., Reading, MA, 1995.